This document is downloaded from DR-NTU (https://dr.ntu.edu.sg)
Nanyang Technological University, Singapore.

Web-based 3D virtual environment for information visualization

Neo, Ker Sin

2005

Neo, K. S. (2005). Web-based 3D virtual environment for information visualization. Master’ s thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/4933

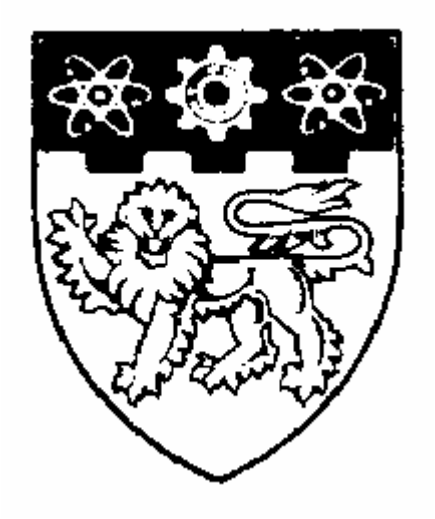
https://doi.org/10.32657/10356/4933

Nanyang Technological University

Downloaded on 09 Apr 2024 14:53:35 SGT

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

WEB-BASED 3D VIRTUAL ENVIRONMENT FOR INFORMATION VISUALIZATION



NEO KER SIN

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

2005

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Web-Based 3D Virtual Environment For Information Visualization

Neo Ker Sin

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2005

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Acknowledgements

I would like to express my heartfelt gratitude to my project supervisor, Dr. **Lin Qingping** for his guidance and patience. Dr. Lin's recommendations and suggestions have been invaluable for this project. In addition, Dr. Lin also enlightened me on research methodologies and proper technical writing. I would also like to thank my co-supervisor Professor **Robert Gay Kheng Leng** for his encouragement and words of advice.

The Managed Computing Competency Centre (MC3) and Information Communication Institute of Singapore (ICIS) has provided me with much needed sponsorship. I would like to express my thanks to the staff at MC3 and ICIS, especially Professor **Robert Gay Kheng Leng** (Director of MC3), Associate Professor **David Siew Chee Keong** (Director of ICIS). Without their support, this research project would not be possible.

I would also like to thank everyone at the InfoComm Research Lab for their friendship and assistance over the last two years. Finally, I would like to thank my parents for their love and support.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Table of Contents

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

List of Figures

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

List of Tables

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Summary

Information visualization is valuable in learning abstract ideas and new concepts. The existing technology for publishing online papers is lacking in terms of user interaction as well as dynamic visualization of abstract concepts. Currently, authors of electronic documents publish their work in textual format with 2D (2-dimensional) graphics. The presentation of electronic documents does not fully exploit what the current technology is able to offer. Often, the visualization techniques employed are static 2D illustrations. 2D visualization is inferior to 3D (3-dimensional) visualization, as it does not provide interactive features like scene navigation, zooming, and object manipulation.

In this project, we proposed a web-based system that allows authors of electronic documents to create their own interactive 3D visualizations. Our proposed behavior definition model allows a content author to model behaviors of virtual objects by providing an interface for interacting with the 3D virtual environment. This improves the usability of the system as users with little or no programming and modeling experience will be able use it with little difficulty.

Furthermore, the proposed framework supports creation of animations and modeling of object behaviors based on mathematical models. Mathematical-based modeling allows more complex object behaviors and interdependencies to be created. Our proposed scripting syntax further improves the usefulness of the system. The proposed system provides a solution for authors of web documents who wish to create powerful visualizations of abstract concepts over the Internet. The 3D visualization facilitates learning and understanding of scientific concepts.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Chapter 1 Introduction

3-Dimensional (3D) animation is an effective mechanism in visualizing scientific concepts and ideas. This is especially true if the presented information contains abstract concepts and ideas.

With the advancements in visualization techniques and the growing popularity of Internet technologies, the Internet has become an indispensable medium for sharing of knowledge. The data presented on the World Wide Web today is no longer limited to textual data, but also includes other formats such as pictures and videos. Researchers can share their ideas and research findings by publishing the results of their work on the Internet. Attaching animations and 3D visualizations to these documents would be valuable in helping readers gain a better understanding of the topic at hand. However, it is currently not easy to add 3D visualizations to electronic documents.

1.1 RESEARCH MOTIVATION

From our survey of the current visualization techniques for online journal (or documents) publication, we observed that more sophisticated mechanisms could be provided for visualizing the information presented in online documents, particularly abstract concepts in scientific journals.

The most commonly used visualization technique is 2-Dimensional (2D) graphics. This can be in the form of a photo or computer generated graphics. While convenient to create and use, this has the disadvantage of being static and non-interactive. Representing a 3D object, such as the structure of a car engine, in a 2D picture also makes it more difficult for a user to fully understand the structure.

Animated scenes/pictures further enhance the effectiveness of visualizing information. 2D animations have been used extensively in visualizing information. While photographs and video-clips are useful in presenting

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

information, it is not always possible to take a photograph or video of the object of interest in practice. For example, it is dangerous, if not impossible, to make a video of the internal activities in a live volcano. Other scenes may be too small to be filmed, for example atomic structures. Video files also consume much precious bandwidth and it is not practical to use them extensively in web applications.

Representing 3D objects in 2D visualization is unnatural. Coupled with the lack of navigational capabilities in generic 2D animations, the users frequently have to mentally visualize the 2D animations in a natural 3D environment. Furthermore, in a 3D environment, users are able to manipulate and examine objects closely; this feature is not present in 2D visualization.

3D visualization is much more powerful and effective than 2D visualization, but many web applications today have not exploited the advantages of what 3D visualization can provide. This is mainly due to the following reasons:

- Large bandwidth consumption when using 3D visualizations over the Internet.
- Some sophisticated 3D modeling and animation tools are available for creating realistic visualization models but they are not suitable for distribution over the Internet.
- Difficulty in learning 3D modeling tools and language.

Thus, the motivation of this research becomes clear – there is a need to enhance the effectiveness of online papers. A tool could be provided for authors of electronic documents to define and create interactive, animated 3D scenes with greater ease. This will be invaluable in visualizing abstract scientific concepts and modeling complex information such as chemical bonds. In addition, the interactive contents will enable users to gain a better understanding of the information modeled.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

1.2 CONTRIBUTION

This project was motivated by the lack of proper visualizations in the publication of online documents. A survey of the current state of technology was then carried out to justify the need for further research.

A novel mechanism that allows content authors to create 3D objects and define interactive behaviors was then proposed. This mechanism maps user-defined data onto the VRML (Virtual Reality Modeling Language) scene, thus providing a user-friendly tool for creating interactive 3D scenes that can be displayed on the Internet.

This project is distinct from other similar work as it incorporates animation and behavior rules in its design. The proposed framework allows users with little or no background knowledge in 3D modeling and animation to create 3D objects, animate these objects, and define their behavior and interactions.

While similar animation and modeling tools exist, they provide limited support for defining interdependencies among objects. Frequently, users have to create time-frame based animation; this results in estimation errors and is inferior to physics-based modeling.

Our proposed mechanism thus aims to improve on these two aspects of similar animation tools by providing support for mathematical model-based animations, as well as providing support for a small scripting language that can be used for controlling behaviors of 3D objects. This mechanism allows authors to create physics-based modeling of animations, as well as define interdependencies between multiple 3D objects using a web-based GUI.

The proposed mechanism can be used on any web application where 3D visualization is desired. Possible applications include the publication of online documents and e-Learning. “Research suggests that using visual treatments in lessons enhances learning with varying degrees of success” [1], thus the

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

proposed mechanism adds more value to web-based e-Learning applications. In addition, online documents where abstract ideas are presented will also benefit from the inclusion of dynamic visualizations that allow user interactions.

1.3 OBJECTIVES

The main research aim of this project is to explore and devise a mechanism for creating interactive visualization of abstract scientific concepts in a web-based 3D virtual environment. More specific objectives are as follow:

- To carry out a critical review of the current state of the art visualization techniques. This gives a clearer picture of the current state of various visualization techniques and justifies the need for further research to be carried out in this area.

- To devise a web-based visualization method. This allows interactive visualizations to be viewed on a standard browser, thus reaching out to a larger audience. This method should allow content authors to create 3D visualizations with interactive behaviors, hence enhancing the usefulness of the visualization. Research effort should be focused on devising a novel method for mapping user-defined animation rules onto 3D objects so that visualization can be achieved.

- To design the framework for the proposed method and to implement a prototype that realizes critical features of the design. The prototype serves as a proof-of-concept for the proposed method.

- To focus on a target audience that consists of document authors with little or no programming background and can create mathematical functions-based modeled animations. This allows a larger group of users to exploit the usefulness of the proposed visualization method, as we aim to make the interface user-friendly for programmers and non-programmers.

1.4 ORGANIZATION OF THESIS

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

This thesis is organized as follow:

- Chapter 2 is dedicated to the literature survey on research work that is related to this project; an analysis and description of the problem that forms the motivation for this project is also given.

- Chapter 3 contains the proposal for a possible solution to the problem highlighted in Chapter 2.

- Chapter 4 contains a brief description of the proof-of-concept prototype that was implemented to justify the feasibility of our proposed framework.

- Chapter 5 concludes this report and gives several recommendations that could improve this project further.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Chapter 2 Literature Review

This chapter gives a critical review of the current state of visualization techniques available; it provides the background and scope of this research. A study and review of visualization is given, with strong emphasis on web-based 3D visualization. Animation and behavior modeling techniques are reviewed. Finally, the need for further research in web-based behavior modeling of 3D visualization is examined.

2.1 VISUALIZATION TOOLS AND TECHNOLOGY

We classify visualization techniques into two broad categories – 2D visualization and 3D visualization. As our research objective is to develop a **web-based** solution for 3D object and behavior modeling, we focus mainly on the review of tools that can create visualizations suitable for distribution over the World Wide Web.

2-DIMENSIONAL VISUALIZATION TOOLS

The Macromedia Flash Player [2] is a popular visualization tool for presenting interactive multimedia on the Internet. Plugins are available for major browsers (such as Internet Explorer, and Netscape Communicator), thus allowing the animations to be viewed within the browsers. Coupled with an effective compression algorithm, Flash is highly valued as a mechanism for publishing multimedia content over the Internet [3]. The real strength of Flash lies in its programming language, ActionScript [4], which allows programmers to create interactive animations.

The limitation of Flash in information visualization is its 2D environment. In a 2D environment, viewers are unable to navigate the scene, manipulate and study objects as naturally as in a 3D environment. In addition, all animations have to be pre-recorded, and not generated ‘on the fly’. Thus, it is not practical for use in truly dynamic visualizations.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

3-DIMENSIONAL VISUALIZATION TOOLS

Using 3D editors, developers could construct realistic 3D models of real-life objects. These 3D editors are excellent for creating virtual models and scenes. With navigational capability, users may navigate through the virtual scene, and observe the objects from different angles. Simulations in virtual 3-D environments provide a more realistic experience than simulations done in 2D animations.

There are a number of 3D editors available for purchase (3D Studio Max [5], Ulead [6]) or free (Blender [7]). Among these, 3D Studio Max is capable of producing movie-quality animation and excellent object rendering. Several limitations make it unsuitable as a format for publishing 3D content over the Internet. Even though 3D Studio Max allows users to export 3D content to a format (VRML) suitable for publishing over the Internet, it is a costly solution that is available only to users who can afford the heavy price tag.

2.2 WEB STANDARDS

Researchers have realized the need for a set of web standards for publication of 3D content over the Internet. A set of well-defined standards that is well-conformed enables the created 3D content to reach out to a larger group of audience. This is because content that conforms to a particular set of well-defined standards can be viewed on any browser that supports the standards. Therefore, a set of well-conformed standards improves portability. In this section, we give an introduction to the four major web standards for publishing 3D content on the Internet.

VRML

VRML, Virtual Reality Modeling Language [8], is a platform-independent modeling language used for creating three-dimensional virtual worlds and objects that can be displayed on the Internet. VRML does more than creating

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

virtual worlds and objects; it also allows users to navigate freely in the virtual world, as well as manipulate objects for closer inspection. VRML may be used in a variety of application areas including engineering and scientific visualization, multimedia presentations, entertainment and educational titles, web pages, as well as shared virtual worlds. These will be further discussed in Section 2.3.

In 1997, the VRML 97 Specifications [9] was released. This brought about major improvements in the language. In VRML 1.0, it was possible to display static worlds; a user can freely navigate in these virtual spaces as well as observe any objects within. With the introduction of VRML 2.0, and later VRML 97, it is now possible to build dynamic, interactive worlds where objects can be animated. In addition, users can also interact with the objects, and experience cool sound effects that are embedded in the virtual environment. Besides representing 3D and multimedia objects, VRML is also capable of hyper-linking them to other media such as text, sounds, movies, as well as images. The VRML 2.0 specs also provide the ability to associate behaviors via JavaScript to various parts of the model. Thus, the model can now respond to user actions (more interactive) and modify itself dynamically.

The introduction of the External Authoring Interface (EAI) allows further interaction with the 3D model [10][11]. Users can now interact with VRML models via an external application, a Java applet embedded in the same webpage. Using EAI, it is also possible to create new objects/animations in a VRML scene via the applet interface.

X3D

For better integration with the next-generation web technologies, which is expected to rely heavily on XML, the W3C is also working on X3D [12]. X3D is the new open standard for 3D content on the web and it is designed to be compatible with existing VRML browsers and tools. X3D can be described as a specification of the VRML scene graph in XML standards, thus it can be easily integrated into applications that support XML.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

X3D is also component-based and extensible; this allows easy extension of features in future. In addition, it is easy to update and preserve VRML content as X3D. With a set of well-specified standards, developers will be able to create more conformant and consistent 3D implementations in future.

JAVA 3D

Java 3D [13] is a set of API that allows developers to create 3D objects and visual environments. The set of API provides object-oriented interfaces that support a simple and high-level programming model. Furthermore, as it is based on the Java technology, any implementation using Java 3D will be platform-independent.

XJ3D

Xj3D [14][15] is an open-source toolkit for viewing or importing content that conform to VRML and X3D standards. This implementation allows developers at the Web3D Consortium to try out new application areas of the X3D specifications, as well as provide a library for application developers to use in their own implementations that support the X3D standards.

OUR CHOICE

VRML was chosen as a 3D modeling language for this project. As our objective was to build a web-based authoring tool, it was important for our implementation to be compliant with current web standards. During the initial development stages of this project, there was no fully functioning X3D browser available. In addition, X3D is designed to be fully compatible with VRML. Thus, our choice of VRML as a 3D modeling language will ensure the extensibility of our work when the X3D browser is ready.

2.3 APPLICATIONS OF 3D VISUALIZATION

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

In this section, we discuss some areas in which the benefits of 3D visualization have been exploited. In the last ten years, significant development in 3D visualization includes research work in the areas of Collaborative Virtual Environments (CVEs) [16][17], visualization of information, modeling in various industries like construction and medical research, multimedia presentations, and e-Learning.

COLLABORATIVE VIRTUAL ENVIRONMENT (CVE)

One of the major areas of research exploiting 3D visualization is Collaborative Virtual Environments (CVEs). CVEs make use of network protocols to implement a large-scale Virtual Environment (VE) where users can access from different geographical locations, and interact with each other. The earlier CVE standards that were developed for use in military applications include SIMNET (Simulator Network) [18], DIS (Distributed Interactive Simulation) [18] .

CVEs have also been developed for leisure and entertainment; these include Diamond Park [19], which was the first DVE to combine visual, audio, and kinesthetic interaction with extendibility and scalability (different from MASSIVE [21]). The users can interact with each other using speech, sound, and animated 3D graphics in realtime. Cyclists in the VE represent visitors; they are able to explore the environment as well as interact with other users. Diamond Park was implemented using the software platform SPLINE [20].

Another significant CVE project is NPSNET [22][23]. NPSNET is used in a variety of research areas including medical and emergency training, networked realtime hypermedia low-cost 3D sounds, simulation-based design etc. Additionally, ongoing research effort is still invested in NPSNET-V [24], which provides a framework for creating virtual worlds, allowing new components to be added dynamically during runtime.

GEOVRML

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Besides CVEs, 3D visualization techniques have been used to model geographical features. GeoVRML [25][26][27] is an extension to the VRML standards. It provides the geosciences field with the ability to model dynamic 3D geographic data that can be distributed over the web and interactively viewed using a standard browser configuration. The Java implementation is released as open-source – interested developers can improve further. In future, diverse applications such as weather simulation, urban planning, virtual real estate, and virtual tourism may be possible with the tools that support GeoVRML. One major drawback is that the size of the files created is usually very large and may not be suitable for transmission over the Internet. Perhaps with the maturity of this technology, our real world can be modeled and used to enhance the 2D satellite images we are seeing now.

CONSTRUCTION AND ARCHITECTURE

A study on the modeling of steel structures and construction equipment as objects in construction-site VE models was done [28]. Object-like VRML representations allow for easy updating and extending implementation details without changing the model itself. Text pop-up allows a user to view the properties of each beam. The user is able to specify the spine and end points of each beam. To improve loading time, trucks, cranes and other equipment were not modeled extensively, unnecessary details were left out. However, since users perform excavation operations through the use of widgets (controlled by mouse movements), this implies that only one widget can be manipulated at one time, unlike a real excavator where most of the controls can be manipulated simultaneously with both hands. In addition, appearance of digging and dumping dirt is only visual simulation; it is not an actual physics-based simulation. A useful improvement would be to devise a system that can provide actual simulation of the work done on a work-site, such as digging dirt off the ground, and moving of gravel across the site. It is also useful to develop a real-time virtual world where changes done at a work-site is updated on its corresponding virtual model.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Another proposed system, OSCONVR (Open System for Construction, Virtual Reality) [29] is an interactive interface to an Object DB for construction architectural design. The novelty of this project is that VRML is intended to be an interface rather than a visualization tool of an integrated project DB using the Internet. This allows users to view and examine 3D models of the different materials used in a building. Navigation is made easier by this visual interface as users can select items based on the 3D models; there is no need to locate items by complicated item numbers or technical terms. This method could be further improved by including a showcase of the different materials that can be used to build a house, and not restricting the user to materials used on the construction of an existing house.

MEDICAL RESEARCH

In the field of medical science, 3D visualization is used for the purpose of education and diagnosis and is becoming increasingly more important. VRML is a good candidate for Web3D as it is popular and easy to manipulate.

Jonghee Han et al. [30] created 3D models of the larynx and displayed them on the web. As the larynx is very complex, it is difficult to obtain a detailed structure by conventional dissection, modeling it in VRML is very helpful for students trying to understand the structure. To make this visualization more realistic, animations can be included to allow learners to see how the larynx actually moves when a person speaks.

EDUCATION

In this section, we discuss how 3D visualization is being used in training and education systems.

When evaluating the effectiveness of using VE to train individuals in navigating within a complex building, it was found that VE rehearsal produced more route knowledge than verbal rehearsal [31]. On the other hand, the route knowledge gained from VE rehearsal was still lesser than that gained from

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

training in the actual building. However, we feel that the sample size of 60 students was too small to make a generic conclusion on the effectiveness of using VEs in navigation training programmes. When using VEs to teach science structure and process, scenario tests from the NDSU World Wide Web Instructional Committee (WWWIC) show that learners are generally as satisfied with their learning experience using VEs as they are with traditional study methods [32].

VRML is useful in the teaching of scientific subjects. Omer et al. [33] described the usefulness of using VRML as a visualization tool in the teaching of scientific subjects. The value of VRML as a 3D modeling language in modeling complex molecular models, chemical reactions (using animations), virtual laboratories, and data mining is recognized. Additionally, Ruiz et al. [34] proposed a model for the development of chemical experiments. In the 3D virtual lab, chemical experiments are defined through the use of a set of materials, which can be manipulated following a set of defined protocols. Although this virtual lab gives some degree of freedom to the user in terms of varying the stages in an experiment, it does not provide any component reuse in the sense that the amount of chemicals used in each method is fixed. For the same experiment with different amounts of chemicals used, a brand new method has to be defined.

The Interactive Universe (iUniverse) [36] project aims to help learners to enjoy learning more about the Universe by modeling regions from the scale of the Earth up to the Universe. Modeled entirely in VRML, the iUniverse can be viewed on web browsers with a VRML plugin installed; users could also navigate within the “Universe” as well as zoom in on “planets”.

VRML could make use of DVEs (Distributed Virtual Environment) to allow students to gather online and learn interactively together [35]. In developing VEs for educational purposed, we agree that it is important to remain focused on the importance of interactivity in learning, and not blindly develop an impressive educational VR system where most students do not have access (because immersive technology requires expensive hardware).

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

The effectiveness of e-Learning is a much-debated one. While some may argue that it is a waste of time [37], others are actively trying to develop better authoring tools for e-Learning systems [38][39]. While static visualizations add value to e-Learning systems, we believe that animations and object behaviors can further enhance e-Learning systems by providing learners with a richer learning experience.

INFORMATION VISUALIZATION

Information visualization can be as simple as a knowledge map scribbled on a piece of paper during a brainstorming process, or complex multi-dimensional views of data. The purpose of information visualization is to present information in a way that facilitates understanding. In this section we review how 3D environments can be used to visualize information.

M. Kreuseler et al. [42] proposed a scalable framework for information visualization that maps points on 2D Cartesian map onto a 3D sphere, thus allowing a multi-dimensional view of the information. By ‘magnifying’ areas of interest and keeping the other regions in normal resolution, users are able to view in greater detail areas of interest, but maintain a perspective of the whole system. However, for a large information space, the processing time is going to be significant as a large number of polygons needs to be generated 'on the fly' to obtain a smooth surface.

Information Flocking [40] is a way of visualizing data based on the schooling behavior of fish. This novel method allows users to see complex correlations between data items through the amount of time each fish spends near others. As fish in the simulation can change their behaviour in response to changes in the underlying data set, the human user is able to identify patterns more easily. This is an example of how object behaviors can be used to convey information. However, for large data sets, the computation time for the fish simulation may be significant.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Information from the Internet could be retrieved and the results displayed in 3D models using VRML [41]. Realistic models provide a more intuitive way of navigating the results. For example, a user can find out more about the structure of a house by navigating around the environment of the house.

We recognize the value of VRML as a modeling language for displaying visualizations. As our research focus is on providing mechanisms for content authors to create behaviors and animations, we will now look at some research work related to animations and behavior modeling in the next section.

2.4 MODELING BEHAVIOR AND ANIMATIONS

Animated visualizations enhance the learning experience and help to improve understanding of a scenario. In this section, we give a review of some research work that has been done on animations and behavior modeling. As behavior modeling is a big area, we focus on projects utilizing web-based technologies (VRML, X3D, XML) as well as projects targetted at users with no programming experience (Alice), and projects that aid learning (algorithm animations).

This section is divided into two sub sections, the first sub section reviews work that is related to VRML, and the second reviews work that has been done based on other technologies.

We first discuss briefly the three paradigms [43] for implementing animations in objects. The first is **keyframe animation**. In keyframe animation, the user defines the initial and final frames of an animation sequence, and a set of interpolated values for the transformation between the initial and end frames. This method does not provide very realistic movements, and can be found in popular tools such as Macromedia Flash. In the **kinematic model**, the movement is defined using an equation (or a set of equations) with kinematic variables[1] as a function of time. Finally, in **dynamic** animations, movement of

[1] Such as position, velocity, and acceleration.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

objects is defined using an equation (or set of equations) with dynamic variables[2] as a function of time.

We can further classify the process of generating each individual frame in the animation sequence – **frame-by-frame** and **real-time** animation [44]. As an animation should be played with at least 18-24 frames per second (for example, movies are played at 24 frames per second in the cinema), real-time animation is not always possible. This is especially true when rendering complex 3-D objects, thus offline frame-by-frame generation ensures better results. Playing animations at a frame rate less than 18 frames per second results in animation that is not smooth.

VRML

Animated VRML objects provide effective visualizations as it allows the user to observe the objects from multiple viewpoints. It also provides a more realistic experience because the user is able to visualize it in three-dimensional space.

Various researchers have worked on animations in VRML. One particular interesting project involves developing an animation toolkit based on motion mapping [45]. This toolkit is outstanding as it is able to map motions of one object onto another different object. For example, it is possible to map the flying motion of a bird object onto a fish object. This is achieved by extracting modeling information separately from animation information. As the main focus of this project is on interactive motion modification and re-usability, accurate modeling is not emphasized. Thus, possible improvements to this system could include more accurate modeling.

Another interesting animation project involves using VRML and Java to construct interactive animations [46], whose behavior is defined in real-time by user's actions. The user of this system is able to define the animation parameters

[2] Such as mass, and force.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

using a Java application. The VRML script node processes the user input before the necessary changes are written to the VRML file. The system lacks the ability to allow users to define their own motions. Users can only choose from a list of pre-defined animations and vary the animation parameters.

Researchers have also worked on extending the VRML specifications to improve the animation and behavior system. VRML++ [47] is an extension of VRML97. It introduces classes and inheritance, an improved type system, and dynamic routing. The authors aim to reuse and parameterize animations and behaviors of VRML objects through the use of generic animation classes that get sensors and interpolators as parameters. VRML++ also introduces the concept of dynamic routing. In a connection class, routes can be created between nodes. These nodes are passed as field values to a prototype of the connection class. Thus, when we need to create new routes, we only need to create a new instance of the connection class and pass in the names of the nodes as parameters. One major flaw of VRML++ is that a preprocessor is required to translate the VRML++ file into VRML97 file so that it can be viewed with every VRML97 browser; this requires working knowledge of VRML.

OTHER TECHNOLOGIES

Other animation management methods include the event-condition-action [48] rule that works in such a way that when an event occurs, and a certain (set of) condition(s) is met, a pre-defined action will be performed. The definition of animations is highly structured and thus easily managed. A generic system that provides interactive real-time animation was proposed by Marita Duecker et. al. [49]. The animation system consists of three main components that are decoupled to facilitate execution in a distributed environment. A maximum degree of interactivity with the user is provided. This is possible as the interpreter is able to receive user input from the editor module while the animator is generating the animations. A dispatcher receives signals from the interpreter and serializes the input requests before passing them to the animator.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

On the other hand, researchers have also been working on developing middleware for users who are inexperienced in 3D modeling. Such applications make it easier for non-programmers to create virtual scenes and behaviors. The CONTIGRA [50][51] architecture enables the construction of interactive 3D applications (standalone or web-based) using declarative XML documents describing the component implementation, its interface, as well as component configuration and composition of 3D user interfaces and Virtual Environments. This architecture separates the description of geometry, behavior, and audio, thus facilitating component reuse. Theoretically, any 3D format is supported as long as the necessary transforming modules are available to transform the XML descriptions into the desired format. However, in practice, this is difficult to achieve as developing transform modules for this purpose is time-consuming and tough.

Behavior3D [52] allows for declaratively modeling of 3D object behaviors. Behavior3Dnode is an XML-based language that acts as an interface definition of new nodes. Based on X3D, Behavior3D uses a node concept that supports object-oriented features like inheritance, strong typing, and polymorphism. Behavior3D has a novel grammar generation mechanism that simplifies implementation; however, the animations defined are time-frame-based.

The researchers at the University of Virginia have been working on and developed Alice [53][54][55], a desktop 3D authoring system. Emphasis of Alice is on authoring the time-based and interactive behavior of 3D objects, rather than on creating geometries. The system was designed for undergraduates with no 3D graphics or programming experience. Thus, mathematical notations are avoided in the syntax as the target audience is typically from non-science/engineering faculties. In addition, the syntax of Alice is easily comprehensible; this coupled with the fact that the system was tested on real users (undergraduates) improves the usability of the system. The object hierarchy system used allows objects to be managed individually or collectively as a group. However, Alice is a standalone application that only runs on Windows Operating Systems, and currently does not support VRML;

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

this makes it unsuitable for developing 3D visualizations that can be distributed over the World Wide Web.

In the W3C, SMIL (Synchronized Multimedia Integration Language) [56] is being developed as a framework for animation support. SMIL incorporates animation onto a timeline and provides a mechanism for composing multiple animations. Animation functions define how an attribute of a particular target element can be changed over time. SMIL also allows animations to be applied collectively to attribute(s) of a particular element over the same period of time. Currently, SMIL presentations can be run on Internet Explorer 5.5 or later as standard HTML files for multimedia presentations such as images, videos, and audio clips.

In addition to generic animation generation systems, studies on algorithm animations [57] have also been done extensively. Algorithm animations aim to positively impact the learning of computer science subjects. Samba [58] is an animation interpreter that reads command lines and performs the corresponding animations. Jsamba [59], the Java version of Samba is available online as an applet; it allows users to type in animation commands and have a feel of how the animation system works. Compared to Alice, the commands for Samba are more difficult to understand, this could be due to the difference in target audience. Whereas Alice was designed primarily for non-programmers, Samba is mainly targetted for users who are more technically-inclined.

JAWAA 2.0 [61] is a scripting language for easy creation of animations on the Internet; it is designed for both novice and experienced programmers. The scripting language can be easily learned by beginners; advanced users can add JAWAA [60] commands to their program to produce animation of data structures. The JAWAA Editor interface allows a user to lay out objects and then modify these objects across time to produce animations; mathematical functions-based definition of animations is not provided.

AnimalScript [62] aims to overcome the shortcomings of JAWAA by focusing on extensibility; this allows features, such as support for animating data

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

structures that are not included in the implementation, to be added easily. The main advantage of AnimalScript is that it can be used for more "generic" animations while other systems such as JAWAA are restricted to a fixed list of supported features.

There are many tools available to help beginners build simple animations (for example, Easy GIF Animator [63]). The most common ones store the animations as animated GIF files. The process typically involves preparing a series of pictures and defining the frame rate for the transition from one picture to the next. It is similar to traditional "flipbook" animation in which pictures appear to be animated when the pages of a book are flipped. Most of these tools simplify the process of creating animations and are useful for beginners. These tools are excellent for creating simple animations, but support for interactivity is lacking.

2.5 THE NEED FOR FURTHER RESEARCH

From the literature survey, we discovered that even though there is a good variety of visualization techniques available today, they are still not widely used in the publication of online documents. This is due to the many outstanding issues that need to be addressed in order to make 3D visualization feasible for online publication.

In existing web-based 3D visualization, content authors need to be proficient in a 3D modeling language in order to create meaningful visualizations. Thus, content authors with little or no programming/modeling experience have neither the means nor expertise to create interactive 3D visualizations for online sharing. This is due to the lack of a tool that is user-friendly and suitable for users without programming background.

Most 3D modeling tools require a significant amount of learning time, with the exception of Alice, as it was designed to be user-friendly and non-technical. However, Alice is not a web-based application and does not support

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

mathematical-based behavior modeling. We aim to design a system that provides the same ease of use but with a web-based tool for greater inter-operability.

Some 3D publishing software is not suitable for creating visualizations that can be distributed over the Internet. There are a wide variety of CAD (Computer Aided Design) tools available but most of these are not in a format that is suitable for publishing on the Internet.

The web standards compliant 3D authoring tool, Xj3D, is still in its early development stages. Although it has a lot of potential in future, an author with little background in the development of web3D may not feel encouraged to try it at this infant stage. Furthermore, end users do not have the technical expertise of developers. An application will be fully utilized only if it is simple enough for non-IT experts.

VRML is capable of creating interactive and dynamic 3D objects and animations. It also allows users to define behaviors via scripts. This feature facilitates the creation of more complex object behaviors and relationships, resulting in a simulation visualized in 3D model. From our survey of VRML research work, most modeling systems using the VRML technology do not provide interface/mechanism for users to create more dynamic models. Users who wish to define more complex behaviors like triggering dependent animations need to do so by coding in the VRML file. Therefore, a more user-friendly approach needs to be provided so that users with no experience in VRML coding can create the animation logics. Furthermore, it is especially important to support mathematical model controlled animation for meaningful scientific visualization.

Thus, there is a need to do some further research so that the full potentials of current 3D visualization technology can be exploited and applied to the publication of online documents. Inspired by the ease of use of the Alice tool, we aim to provide a user-friendly interface to VRML with mathematical model-

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

based animation support; this allows content authors to build visualizations and model behaviors with less difficulty than VRML coding.

Thus, the animation goal of the thesis is to provide support for generating animations based on mathematical time-based functions. The visualization goal is to provide a simple mechanism for defining object behaviors for displaying a VRML environment.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Chapter 3 Web-Based 3D Visualization System

In this chapter, we present an interface to VRML. This mechanism allows a content author to create 3D Objects and mathematical functions-based Animations. Our proposal also includes a small scripting language that can be used to define Object behaviors. The proposed mechanism hides the complexities of modeling from the content author; it is thus possible for authors with no prior modeling/programming experience to create meaningful and dynamic 3D visualizations.

3.1 PROPOSED SOLUTION

As mentioned in Section 2.2, VRML will be used as the 3D modeling language in this project. A user interface can be created to allow an end user to input definitions of objects and their behaviors. Using the External Authoring Interface (EAI), we can process these definitions and map them onto the 3D scene. This allows the end user (content author) to create 3D objects and control their behavior without having to learn the 3D modeling language.

Our focus is on providing a convenient 3D authoring tool for content authors to create dynamic 3D visualizations for their documents. While keeping in mind the features available in the authoring tools available, we also aim to add more functionality in order to add value to the proposed system. From the user's perspective, the proposed system should provide an authoring tool with the following features:

- Creating 3D Objects.
- Defining animations using formulae.
- Defining behaviors using a simple scripting language.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

With these basic goals, we aim to provide a tool that is more suitable for authors with limited programming and 3D modeling knowledge, but with knowledge on how to define time-based mathematical functions.

3.2 CORE FEATURES

The main research objective of this project is to explore and devise a mechanism for creating interactive visualization of abstract scientific concepts in a web-based 3D virtual environment. The core features of the proposed system help achieve the objective by providing a user-friendly interface for authors to create and model 3D Objects, as well as define behaviors.

There are two distinct groups of target users – content authors, and content readers. Content authors create the interactive 3D scene and use it to enhance their online publications. Content readers are end users who read these online publications and view the 3D visualizations. From this section onwards, content authors shall be referred to as authors, while content users are referred to as readers.

CREATE OBJECTS

Through the use of a user-friendly interface, the author should be able to create 3D Objects with ease. Knowledge of VRML programming is not required. The user interface should be simple to use even for authors who are creating 3D objects for the first time. More advanced authors can create complex or composite Objects by grouping several basic Objects together, or by importing their own VRML Objects. When Objects are imported into the scene, the author should be able to apply Animations and Behaviors on it.

The creation of Objects should cater to both experienced 3D content authors as well as authors who have little background in programming and 3D modeling. In this way, the system can be used by content authors with different levels of

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

expertise; it will also be sufficient to create visualization of complex scientific concepts such as chemical structure and bonding.

CREATE ANIMATIONS

The interface should allow an author to create meaningful animations with ease. It should not involve complex steps. As discussed in Section 2.4, keyframe animation does not provide very realistic movements. Thus, we have decided to use the kinetic model, where animation parameters are described as a mathematical function of time. This has two distinct advantages – it creates more accurately modeled animations to support visualization of more complex concepts or processes that are not linear in nature; it can also be used by anyone with a simple understanding of mathematical functions.

More importantly, support for mathematical function-based animation is lacking in VRML. To create one, the author has to create scripts that calculate the attribute value at each time instant, and then route these values into the correct Interpolator nodes in order to generate the animation. The proposed mechanism should allow an author to create mathematical function-based animation easily by simply providing a mathematical equation. Generation of intermediate values will be handled by an interface layer that sits between the author and VRML; this should be transparent to the author.

DEFINE OBJECT BEHAVIORS

Finally, the author should be able to define specific behavior for objects. To create truly meaningful visualization systems, one of the core features of the proposed system should be the mechanism for supporting behavior based on the Event-Condition-Action rule.

By supporting this mechanism, more dynamic visualizations can be created. The visualization will consist of more than just a simple sequence of animations; certain actions will only be triggered by special events and only

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

when specified conditions are met. For example, when a proton is fired at an atom, a reaction only occurs when the velocity is sufficient.

We propose a small scripting language. Using this scripting language, the author should be able to specify conditions and the resulting actions. Our proposed mechanism hides the details of conditions checking and sending of events to the VRML nodes for generation of actions (change in state or animation). Unlike using VRMLscript, the author does not need to create procedures, get the specific field values, declare new values, and route new values to the respective nodes in our proposed approach. Thus, the author can focus on modeling the behavior rather than on how to achieve this using VRML and VRMLscript.

3.3 OVERVIEW OF PROPOSED SYSTEM

To provide the desired mechanism for supporting the features described in the previous section, we propose an interface component to VRML. This interface component sits between the user and VRML browser, thus allowing the author to define mathematical function-based animations by entering a mathematical function, as well as define controlled behaviors using a small scripting language. More importantly, this component processes the information entered by the author and communicates with the VRML scene to generate the desired animations and behaviors; this achieves our objective of allowing authors with little 3D programming experience to define animations and behaviors of 3D Objects for visualization over the Internet.

All the Java components will be embedded in the same webpage as the VRML scene. We first introduce the two main controllers – the **Animation Controller**, and the **Behavior Controller**.

The Animation Controller generates the keyvalues for Animations based on the mathematical functions defined by the author. These keyvalues are then sent to the target 3D Object via ROUTE statements. The Behavior Controller parses

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

the script entered by the author. When the event that triggers the particular script occurs, the Behavior Controller gets the latest state information from the VRML scene, and then checks if conditions have been met. It then carries out the actions specified by the author. If the action consists of animation(s), the Animation Controller will be invoked to generate the animation(s).

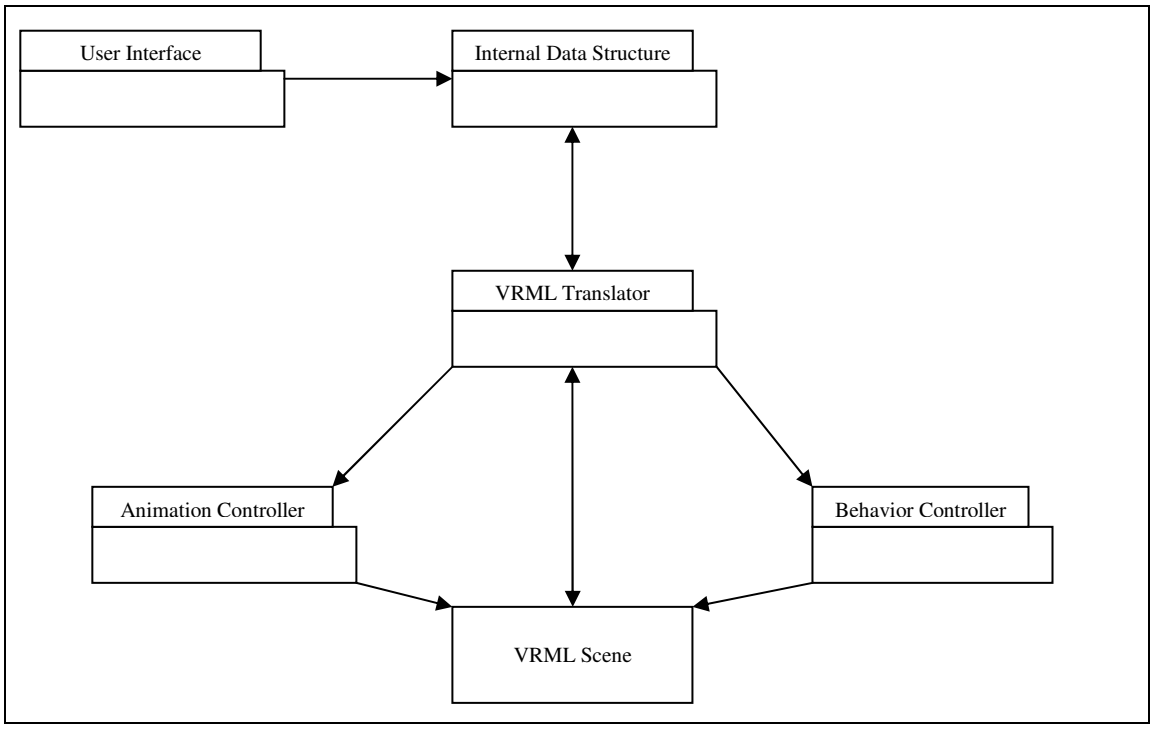


Figure 1 High-Level View of Main Components

Thus with the Animation Controller and Behavior Controller, details of interacting with the VRML scene is transparent to the author. The author only has to focus on modeling the animations, and defining the behaviors, instead of spending time on learning about the mechanisms of generating animations, and checking of conditions in VRML.

The third core component is the **VRML Translator**. The VRML Translator reads the information entered by the author and maps them onto VRML nodes to display the 3D Objects; it is essential in creating the 3D scene as the scene definition will not be stored in VRML format. In the following sub-section, we discuss, in greater detail, how information entered by the author is stored, and explain why this information is not stored in VRML format.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

STORING PROCESSED INFORMATION

As we are proposing an interface to VRML, the information defined by authors will be stored in an external format. This information will be processed by the VRML Translator and generated into 3D Objects in the VRML scene. Animation and behavior definitions will be processed by the Animation Controller and Behavior Controller respectively.

XML is a meta-language for describing markup languages such as HTML. It facilitates the creation of user-defined tags and the structural relationships between them. The semantics of an XML document is either defined by the application that processes the data within or by stylesheets. It is thus suitable for storing animation logics and object data as the structure is user-defined, and the module that processes the definitions decides the semantics.

For this project, we decided to store the data in XML files so as to facilitate data sharing among the different modules; in other words, data portability is ensured. Portability is an important consideration, as the definitions may be used by different components in future prototypes.

DATA FLOW

Figure 2 shows the proposed data flow among the components. The GUI accepts user input and commands. The user input will be parsed and used to generate the detailed data structure. This data structure will be displayed in a Tree structure on the GUI. A Tree structure clearly displays the Objects and Animations defined by the author. The author is able to select an Object or Animation for editing or deletion by clicking on the tree node representing it. Similarly, when the author clicks on an Object in the 3D scene, its corresponding tree node will be selected; this allows the author to select the Object for editing by clicking on its 3D visualization.

When the webpage containing the VRML scene and Java applet is loaded, an XML parser will be used to process the data from the XML file in order to

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

create the VRML scenes. When saving the author's scene definitions to the server, the same module will generate an XML file based on the data structure.

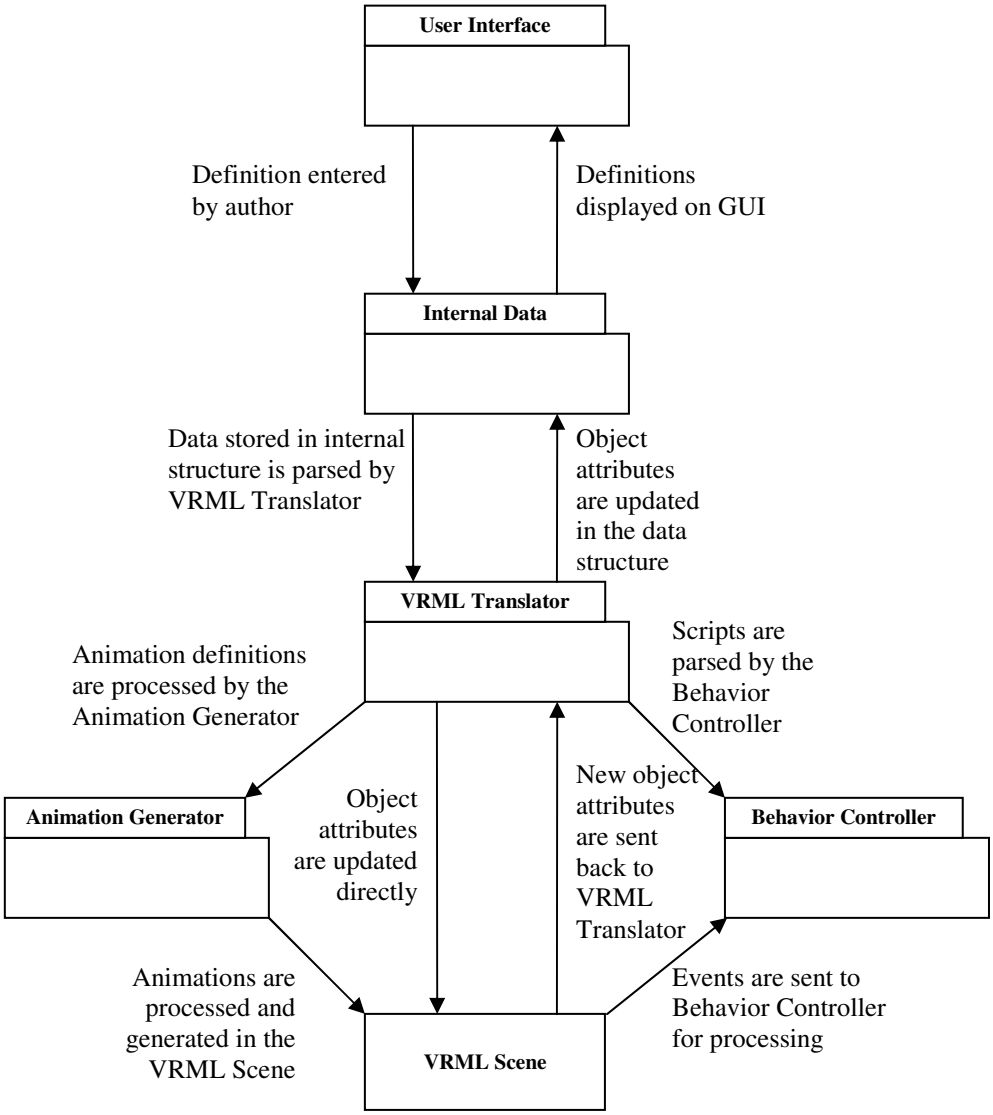


Figure 2 Proposed Data Flow

In the previous section, we discussed the 3 core functionalities that should be provided to the author. Figure 2 highlights how these 3 functionalities can be achieved. Firstly, the system should provide mechanisms for authors to input their 3D object definitions. Secondly, the system should parse the author's input and generate a meaningful data structure. This data structure will be transparent to the end user (author or reader) and will be used by one or more of core modules to generate the VRML scene. Finally, the system should be able to

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

retrieve the data defined by the author and generate it into a VRML scene that contains interactive, dynamic objects.

3.4 DATA STRUCTURE

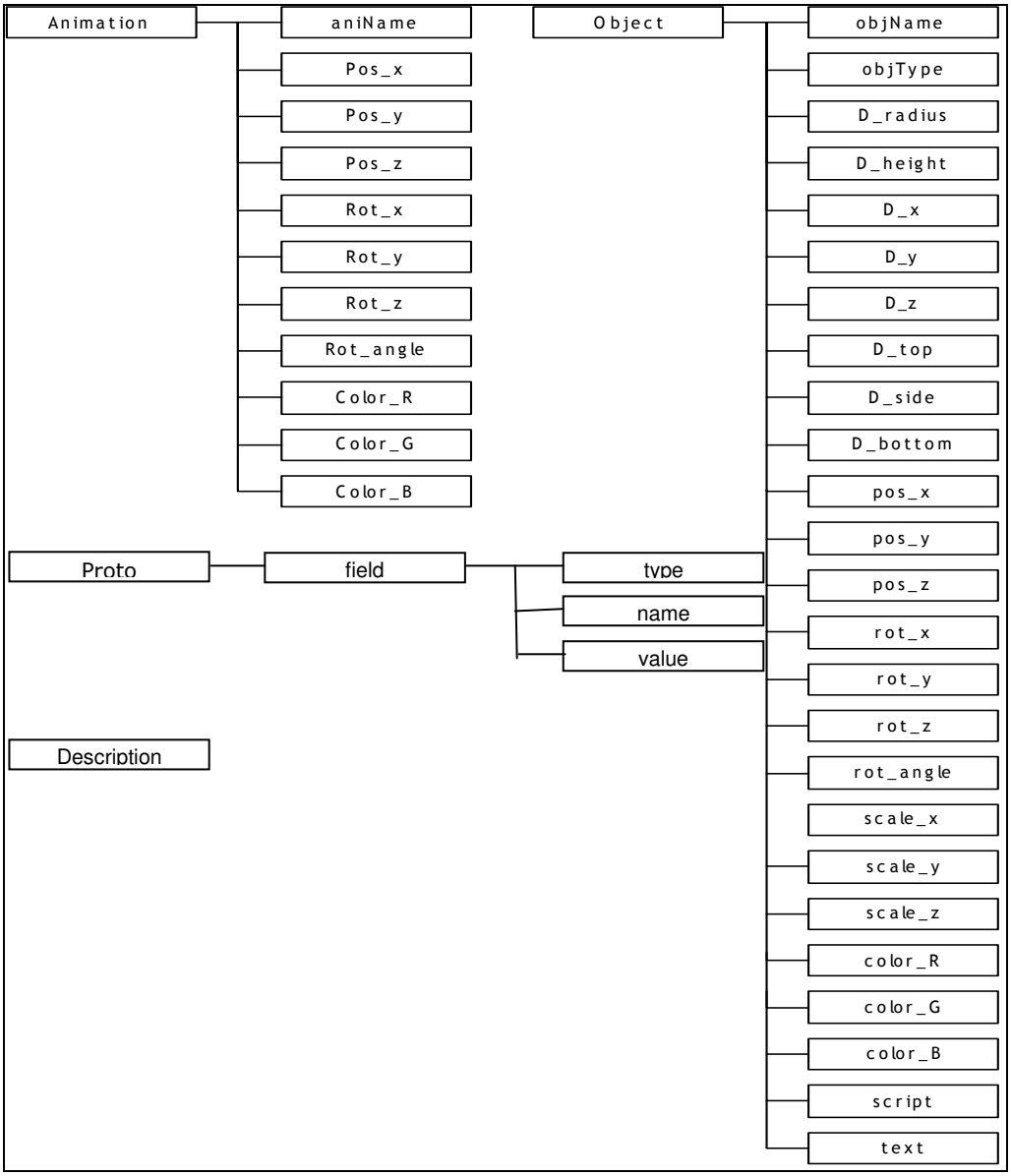


Figure 3 Data Structure of Object and Animation Definitions

The data structure of a scene is shown in Figure 3. From the initial hierarchical structure that is similar to VRML, we have simplified the data structure to a two-layer structure. As the module that handles this information will process the data that is defined, it is not necessary to display the full VRML scene tree

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

to the author. In other words, we only show the author what he is required to know, the implementation details will be invisible to him.

Adopting this data structure also implies that the data processing module has to map the definitions to their corresponding VRML nodes.

Figure 3 also shows all the attributes belonging to an Animation or Object that can be defined by an author.

An Animation definition allows an author to control how an Object's attribute(s) (e.g.: position, orientation etc) changes with respect to time. These can be defined as constants (in the case of linear animation) or as a function of time (in the case of nonlinear animation). Table 1 in Section 3.5 shows a list of supported functions. These functions can be used in combination to produce complex animation controls. When an Animation is applied to an Object, the Object's attribute(s) will change according to the definitions in the Animation.

An Object definition allows an author to define static attributes of an Object such as its size, initial position, orientation, and color.

The 'script' attribute of each Object allows an author to define the actions that follow when that particular object has been clicked. This allows interactive and dynamic behaviors to be created. More details on the scripting syntax will be discussed in Section 3.6.

The Scene Description enhances a reader's understanding of the Scene. An author may add more than one Description to a Scene. Each Description may be used to describe the features and functions of an Object in the Scene. This allows the reader to have a clearer understanding of the Scene.

Finally, to allow extensibility, the data structure will also support PROTO nodes in addition to the basic VRML geometry nodes. A PROTO statement defines a new node type, as well as exposed fields. Providing support for the PROTO node has the following advantages:

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

- **Re-use of elements**. Prototyping allows an author to reuse the definition for an Object to create many instances of the Object. For example, to create a crystal lattice consisting of many repeating units, the author only has to define one basic unit and create many instances of each to form a larger lattice.

- **Easy customization**. As the exposedFields of each PROTO is user-defined, it can be used to customize instances of the same prototype. For example, to create a different crystal lattice, the author can specify different bond lengths and bond angles for each unit. Whenever a unit is instantiated, the author may customize the attributes, thus using one main definition, the author is able to create many different versions of the crystal lattice. Most importantly, the author does not need to create each individual crystal lattice by grouping smaller elements and arranging them into the desired shape.

By providing support for the PROTO node, the system will be able to support a variety of user-defined nodes. This allows more complex Objects to be created. Furthermore, a larger variety of Animations can be defined as the exposedFields of a PROTO node are user-defined. For example, the author will be able to animate different attributes of an Object, such as shape and texture.

3.5 ANIMATION DEFINITION

In VRML, keyframe animation is used. In order to define the parameters for an animation, the author has to calculate the keyvalues at each time instant and enter them in the VRML "Interpolator" node fields. This has two distinct disadvantages. Firstly, the animation being modeled is less accurate when the keyvalues are typed in one by one as values may be rounded off and human errors may occur. Secondly, this mechanism does not respond well to changes in animation rules. For example, when the path of motion of an object changes, the author has to recalculate the keyvalues and edit the VRML file.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Each animation specifies how an Object's attribute(s) change(s) with respect to time. To allow a more accurate modeling of the change, the animation will be defined in terms of mathematical functions in our proposed method. By providing mathematical-based modeling, we allow physics-based modeling as opposed to frame-based modeling; this also saves the author the hassle of defining each value at a specific time instant, as the values will be calculated dynamically.

Table 1 shows the list of mathematical functions that can be used to describe time-based behavior. For example, they can be used to describe how an Object's position changes with respect to time. These mathematical functions may be used alone or combined to create meaningful modeling equations.

Table 1 Animation Definitions

Mathematical Functions	
acos(a)	Returns the arc cosine of a in the interval [0, π] for a in the interval [-1, 1].
asin(a)	Returns the arc sine of a in the interval [0, π] for a in the interval [-1, 1].
atan(a)	Returns the arc tangent of a in the interval [0, π] for a in the interval [-1, 1].
cos(a)	Returns the cosine of a in the interval [-1, 1] for a in the interval [0, π].
sin(a)	Returns the sine of a in the interval [-1, 1] for a in the interval [0, π].
tan(a)	Returns the tangent of a in the interval [-1, 1] for a in the interval [0, π].
exp(a)	e^a, where e is the natural constant 2.718.
log(a,b)	Returns the logarithmic function of a, base b.
lg(a)	Returns the logarithmic function of a, base 10.
ln(a)	Returns the logarithmic function of a, base e.
power(a,b)	Returns the result a^b.
mod(a,b)	Returns the integer remainder after dividing a by b.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Arithmetic Operations	
+	Adds two elements together.
-	Subtracts the element on the right from the element on the left.
*	Multiplies the two elements together.
/	Divides the element on the left by the element on the right.
Others	
()	Operations within brackets will be processed first.

3.6 BEHAVIOR DEFINITION

In this section, we describe our proposed script syntax and justify its value by comparing it to scripting in VRML.

SCRIPTING IN VRML

```
Script {
       exposedField MFString url           []
       field        SFBool   directOutput  FALSE
       field        SFBool   mustEvaluate  FALSE
And any number of:
       eventIn      eventTypeName eventName
       field        fieldTypeName fieldName initialValue
       eventOut     eventTypeName eventName
The actual scripts:
       url "javascript:
       function functionName(parameter) {
              node.set_attribute = TRUE;
       }"
}
```

Figure 4 Script Syntax – VRML Scripting

Figure 4 shows the structure of a VRML Script node. To activate the Script node, a ROUTE statement must be used to send an EventIn to the Script node. Next, the function is invoked and the statements are executed. Finally, the EventOut of this Script is then ROUTE-ed to a target node.

The three major disadvantages of using this are as follow:

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

- The syntax appears confusing to authors who are not experienced in programming. In addition to defining fields and Events for each Script node, the author also has to be familiar with the data types in VRML.

- ROUTE statements have to be used in addition to scripting so that events can be routed to Script nodes and activate these nodes. For an author who is not experienced in programming, this task is intimidating as he has to ensure that the data type of source and destination events in each ROUTE statement match.

- In VRML, an Object sends out an EventOut whenever one of its states changes. For example, when the diffuseColor of an Object changes, a diffuseColor_changed event will be generated. This allows the author to keep track of changes in the scene. In other words, to get an Object's state (such as position), the author has to assign variables to hold these values.

To overcome these disadvantages, we propose a small scripting language that allows the author to focus on the logics of the behavior definition, while the Behavior Controller module works on communication with the VRML Objects. This eliminates the need to understand and use the seemingly complex ROUTE statements, EventIns and EventOuts, as well as fields in scripting.

OUR PROPOSED SYNTAX

Figure 5 shows our proposed script syntax. It uses the most basic **IF-ELSE** construct to implement flow control. An informal survey with students from non-technical faculties revealed that the **IF-ELSE** logic flow was simple enough for most of them to understand. We will discuss this syntax in greater detail before listing out its main advantages.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library


```
IF ( condition(s) )
{
    statement1;
    statement2;
    … ;
}

ELSEIF ( condition(s) )
{
    statement1;
    statement2;
    … ;
}
ELSE
{
    statement1;
    statement2;
    … ;
}

ENDIF
```

Optional

Figure 5 Script Syntax – IF-ELSE Syntax

CONDITIONS

The author may specify one or two conditions for each **IF** or **ELSEIF**. The syntax for conditions is shown in Figure 6.

```
Syntax:

<objectName>.objAttribute=='Attribute-Value'

Examples:

<objectName>.position=='x,y,z'
<objectName>.rotation=='x,y,z,angle'
<objectName>.color=='r,g,b'
```

Figure 6 Syntax for Conditions

Conditions that can be checked include one or more Objects' attribute values at the instant the *current* Object is being clicked. The Object(s) whose attributes are checked may or may not be the Object that is being clicked (*current* Object).

When specifying the position, the values of the x, y, and z components have to be specified, and separated by commas. When specifying rotation, the axis of rotation, together with the angle of rotation, has to be specified. The axis of

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

rotation is the straight line drawn from the origin to the point indicated by the values of x, y, and z.

The color attribute of an Object is determined by the values of the three primary colors – Red, Green, and Blue. The legal values must lie within the range from 0 to 1, for example, '0.2, 0.5, 0.9'. For the scale attribute, the scale factors for the x, y, and z directions in the Object's local coordinates have to be specified.

When two conditions are defined, the author has to specify whether these conditions are required to be an union (||) or intersection (**&&**). In other words, the two conditions must be satisfied at the same time (intersection) or at least one condition must be satisfied (union).

STATEMENTS

Any number of statements may be specified for each **IF**, **IF-ELSE**, or **ELSE**. Figure 7 shows the sample syntax that can be used for the statements.

```
Examples:

<objectName>.position='x,y,z';
<objectName>.rotation='x,y,z,angle';
<objectName>.color='r,g,b';
<objectName>.scale='x,y,z';
<objectName>.animationName='true';
<objectName>.animationName='loop';
```

Figure 7 Syntax for Statements

Each statement has to be ended with a semi-colon (';'). A statement may change a *target* Object's position, rotation, or color attributes. It may also assign an animation to the Object, by specifying the Animation name, and setting it to 'true'.

The syntax for setting the attributes of an Object is the same as that for checking the attribute values of the Object. For each statement, the name of the Object is specified, followed by a dot (.), and then the Object attribute (position, rotation, color, or scale etc). The new values of the attribute are then specified on the right side of the equal (=) sign.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

The author may also assign an animation to a target Object by specifying the name of the Animation and setting it to 'true'. To handle looping in an Animation, the animation property of an Object can be set to 'loop'.

For PROTO nodes, the attribute that can be updated depends on the exposedField defined in the PROTO node. Similarly, Animations can be applied to user-defined fields of PROTO nodes as well. This creates a more dynamic and complex 3D scene.

DISCUSSION

The advantages of our proposed script syntax over scripting in VRML are discussed below:

- Simple but powerful logic flow. Using the most basic form of **IF-ELSE** construct makes the syntax simpler and thus easier to understand. However, this does not compromise its strength in modeling complex behavior relationships. Using the basic construct, authors are able to check for multiple sets of conditions and specify unique actions that should occur for each set of conditions.

- This script enhances the capabilities of VRML. In VRML, animations are generated by events. The occurrence of one event triggers the start of one or more animations. To check for conditions before deciding on the animation to trigger, the author has to use VRMLScript. This method is more tedious and intimidates users who are not familiar with VRML. Thus, without providing an interface for authors who are not experts in VRML scripting, the full capability of VRML cannot be exploited.

- While we provide the mechanisms for exploiting strengths of VRML scripting, the need for knowledge of ROUTEs, Interpolators, and VRML Script is totally eliminated. This results in a much easier way to describe behaviors and animations. For example, authors can directly describe the conditions by specifying the Object name, attribute to be checked, and the value. The Behavior Controller will get the current state of that Object and

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

check it against the value stated by the author. An Object's state can be changed in one statement. The author does not need to bother with data types, ROUTE statements, and events.

Therefore, with our proposed scripting language, the complexities of checking for conditions and generation of new animations are hidden from the author. By providing this additional interface to VRML, we preserve and enhance the capabilities of VRML, while maintaining its usability by authors with varying degrees of programming expertise.

3.7 VRML SCENE GENERATION

The tree structure defined by the author will be generated into an XML file. This data structure will be hidden from the end users (authors and readers). A sample of how an Animation structure would look like is shown in Figure 8.

Since the data structure is slightly different from the node structure in VRML, when converting the data entered by the author to VRML scenes, additional processing needs to be done in order to map the information onto the desired nodes.

In Figure 8, the XML file contains one Animation definition. Any Object that is assigned this Animation **SineCurve** will move along a sine curve, with the x-axis as the time axis. This simple animation example shows the reader the path of a sine function. To illustrate the effect of changing the magnitude and period of the sine function, the author may modify the values of **a** and **b** in the <Pos_y> definition **a*sin(b*t)**.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library


```
- <Animation>
        <aniName>SineCurve</aniName>
        <Pos_x>2*t</Pos_x>
        <Pos_y> sin(2*t)</Pos_y>
        <Pos_z>0</Pos_z>
        <Rot_x>0</Rot_x>
        <Rot_y>0</Rot_y>
        <Rot_z>0</Rot_z>
        <Rot_angle>0</Rot_angle>
        <Scale_x>0</Scale_x>
        <Scale_y>0</Scale_y>
        <Scale_z>0</Scale_z>
        <Color_R>0</Color_R>
        <Color_G>0</Color_G>
        <Color_B>0</Color_B>
</Animation>
```

Figure 8 Sample Animation Definition

In VRML, changing attribute values (for example, an object's color) at each timeframe, and then interpolating these values to create a smooth transition from one value to another, creates an animation. Thus for each function defined, the keyvalue for each timeframe will be calculated, and then mapped onto its respective VRML Interpolator node. By allowing authors to define attribute values based on time, the hassle of having to calculate each position accurately at each time instant is eliminated. This improves the usefulness of the application.

When applying an Animation to an Object, the value of an attribute at the time instant is added to the initial value of the same attribute. For each value of **t**, new values of **x** will be calculated based on the function (2***t**) and then added to the initial value of **x** (5); this gives the value of **x** at each new value of **t**.

For Object definition, the VRML code is generated after the VRML Translator parses through the data definitions. The VRML Translator module that is parsing the XML file knows the semantics of the data; it is thus able to map the data onto the correct VRML nodes for generation of 3D Objects.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library


```
do_extractDefinitionsFromXML;
do_addObjectToScene;

do_extractDefinitionsFromXML
begin
       do_parseXMLFile;
       while(notEndOfFile)
       {
              if (isAniDef)
              {
                     do_instantiateAniInfoClass;
              } else if (isObjDef)
              {
                     do_instantiateObjInfoClass;
              }
       }
end

do_addObjectToScene
begin
       do_createVRMLString;
       do_createVRMLFromString;
       do_extractNewVRMLNodes;
       do_createEventIns;
       do_createEventOuts;
end
```

Figure 9 Mechanism for VRML Scene Generation

Figure 9 shows the pseudo-code for generating VRML Objects from the XML definitions. During the initial page load, the XML file containing the scene definitions is parsed and extracted. The Animation and Object definitions are stored in their respective java objects. The Objects are then added to the VRML scene via the createVRMLFromString() method. After the Object is added, the newly added VRML nodes are then extracted and their respective EventIns and EventOuts created to allow for future communication between the Java Applet application and the VRML scene. The source code can be found in Appendix A.

3.8 ADVANTAGES OF THIS PROPOSAL

This proposal presents three distinct advantages over other similar work:

Firstly, it was designed to be user-friendly. Thus, a simplified user input mechanism was proposed. This allows authors with little or no experience in 3D modeling to learn the basic functions of the system with less difficulty. In addition, the complex mechanism of animation generation, VRML scene interfacing is hidden from the author. Therefore, the author only has to focus on

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

defining the object properties and behaviors without bothering about these details. While Alice is an excellent 3D authoring tool for novice programmers, the objects created are saved in its proprietary format. To view these 3D objects on the WWW, a reader has to download and install the plugin. We feel that saving these objects in a format based on Web Standards would be a better solution as it allows for greater accessibility.

Secondly, unlike other VRML tools, this proposed system allows authors to define animations and dependencies among 3D objects in a virtual environment using **IF-ELSE** logic flow. Even though the script syntax is basic, it is powerful enough for creating well-structured and complex animations for scientific visualization with the support of the **Event-Condition-Action** rule. Furthermore, the use of mathematical functions-based modeling allows more accurate animations to be created without the need for authors to calculate each keyvalue manually.

Thirdly, when defining a formula-based animation, the author simply has to enter a time-based function of the attribute. For example, in Figure 8, to make the object oscillate along the y-axis, the author simply has to input the formula **sin(2*t)** in the field for y-attribute. To create the same animation in Flash, the author has to calculate the values of y at each time interval, and carefully create keyframes along the timeline before creating “motion tween” to generate an animation.

Finally, the definitions will be stored in an XML file. This improves its portability, as any module that knows the semantics of the document is able to process the definitions. Storing definitions in XML file also improves computing performance, as VRML files tend to get bulky as the scene becomes very large and complex. More importantly, the structured nature of XML file enables us to store animation definitions easily and generate the keyframes dynamically when needed. However, there will be a trade-off in processing time that is required to parse the XML file and generate the scene. Even then, we feel that this solution is more bandwidth-efficient as compared to transmitting VRML files directly.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Thus, this proposed solution addresses the issue of providing a user-friendly tool, which is capable of producing dynamic 3D visualizations, for authors with little programming experience.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Chapter 4 System Prototype

In this chapter, we describe the design and implementation of a proof-of-concept prototype that was built to verify the value of our proposal. We first discuss the External Authoring Interface (EAI) and how it helps to achieve our objective of providing an external interface to VRML. Next, we describe the event flow in the client application, as well as highlight the collaboration between the major components in the proposed system. Finally, we present the author and reader interfaces; we also use a simple case study to illustrate the basic steps in using the prototype to create animations and event-triggered actions with conditions. A discussion on the prototype concludes this chapter.

4.1 EAI AND JAVA

The External Authoring Interface (EAI) provides mechanisms for Java applications to communicate with VRML scenes. To the Java application, the EAI is a set of classes that provide methods for controlling the VRML scene; to the VRML scene, the EAI is another mechanism for sending and receiving events. The EAI is a layer that sits between the VRML and Java applet as shown in Figure 10.

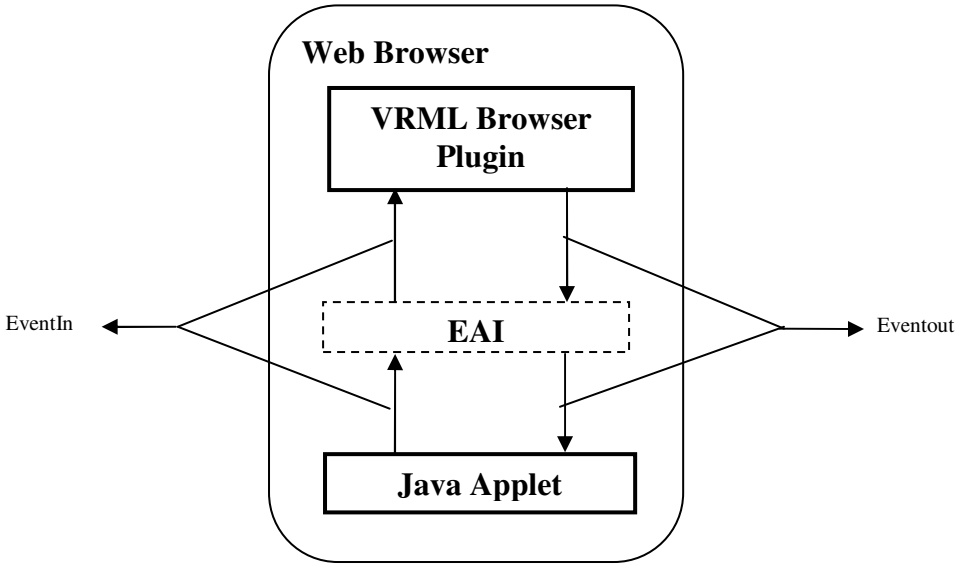


Figure 10 The EAI

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Using a Java-based GUI, we allow authors to define Animations and Behaviors. These definitions will be processed by Java components; events will be sent to the VRML scene to control the behavior of the Objects. When an event occurs in the VRML scene (for example, an Object is being clicked on), the event is sent to the Java components for processing via the EAI. Thus we are able to get the latest states from the VRML scene and process them; these backend activities will be totally transparent to the author.

The EAI provides us with a mechanism for creating communication between external application and VRML scene. With EAI, we can achieve the objective of providing an interface to VRML, hence allowing authors who are less experienced in programming to create interactive and dynamic visualizations in VRML. In the implementation of our prototype, the three major components introduced in the previous chapter (Animation Controller, Behavior Controller, and VRML Translator) process the definitions entered by the author and generate the necessary information required for controlling and communicating with the VRML scene.

Running as a Java applet, the prototype is accessible on web browsers that have the necessary Java and VRML plugins installed. It is thus not restricted to a particular Operating System. In addition, the Java application can process the XML scene definitions before generating the VRML scene via the EAI. Furthermore, authors do not have to install any programs in order to use this tool. Lastly, it is web-based and can thus reach out to a larger group of users situated in different locations.

4.2 THE ARCHITECTURE

The proposed web-based 3D authoring system consists of the following components:

- A webserver computer and one or more client computers.
- XML files describing the visualizations.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

- A server program running on the server computer that receives XML files submitted by client computers.
- Server-side scripts that handle client requests such as file retrieval and updating of information.
- Client-side application (Java applet) performing the following functions:
  - Downloading the scene definitions (XML) from the server, and sending updated scene information back to the server.
  - Processing modules that parse scene definitions and generate information to be mapped onto the VRML scene.
  - Translator module that communicates with the VRML scene via EAI.

The core functionalities described in Section 3.2 will be implemented on the client-side application.

A webserver hosts the files accessible by client machines via a web browser. The client application is a java applet that is downloaded and runs on the client machine. XML files that describe the visualizations will be downloaded to the client machine as well, and processed on the client machine to generate the 3D scene in the browser.

A separate server program was implemented to handle the uploading of XML files submitted by client machines. This program listens for incoming file update requests and saves the uploaded files in the correct directory. Server-side scripts then update the database on the incoming files.

This simple client-server architecture is sufficient to support our requirements for a web-based platform-independent authoring environment. Furthermore, it is a very well-established architecture that supports multiple protocols of communication between the server and client.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

4.3 SYSTEM IMPLEMENTATION

This section describes the implementation of the proof-of-concept prototype.

4.3.1 FLOW OF EVENTS

This section describes the flow of events when a page containing visualization is loaded and when a user (author or reader) clicks on an Object in the Scene.

When the webpage containing visualization is loaded, the XML file containing the Object and Animation definitions is parsed and information stored in temporary data structures. From these data, VRML nodes are generated and added to the VRML Scene to create the necessary 3D Objects. The flow is shown in Figure 11. After the scene is loaded, the author may proceed to add new Objects, Animations, and Behaviors to the scene. As the author is required to define Behaviors using the proposed scripting language, the Behavior Controller will check for syntax errors before saving the changes.

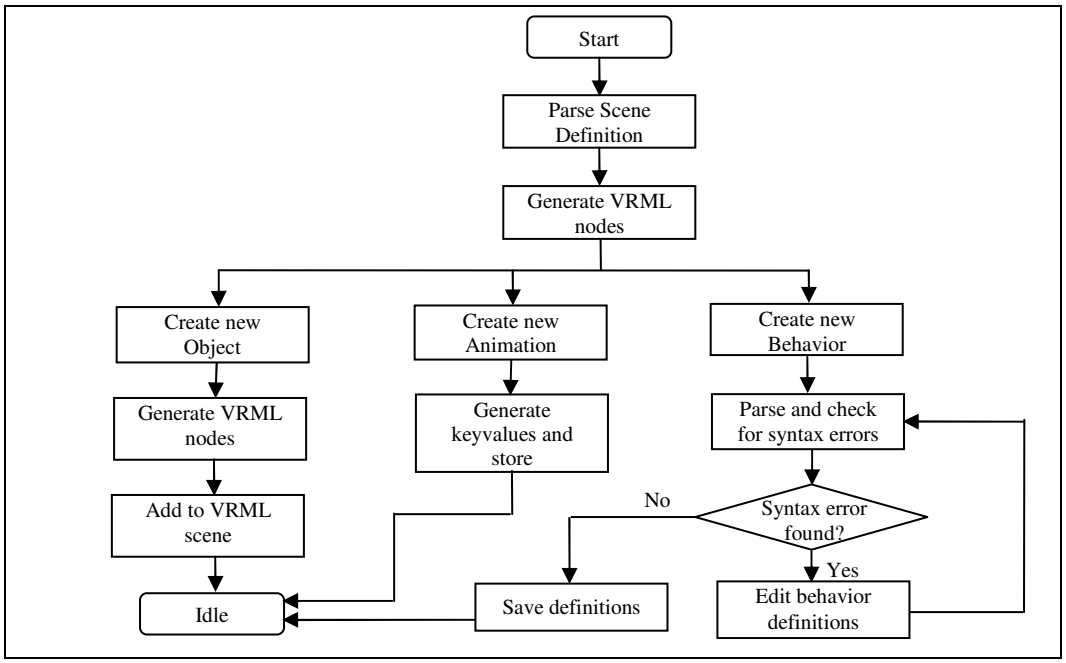


Figure 11 Flow of Events at Page Load

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

At any time, the author may click on the Objects in the VRML scene to trigger some pre-defined actions. Figure 12 shows the events that take place when an Object is being clicked on.

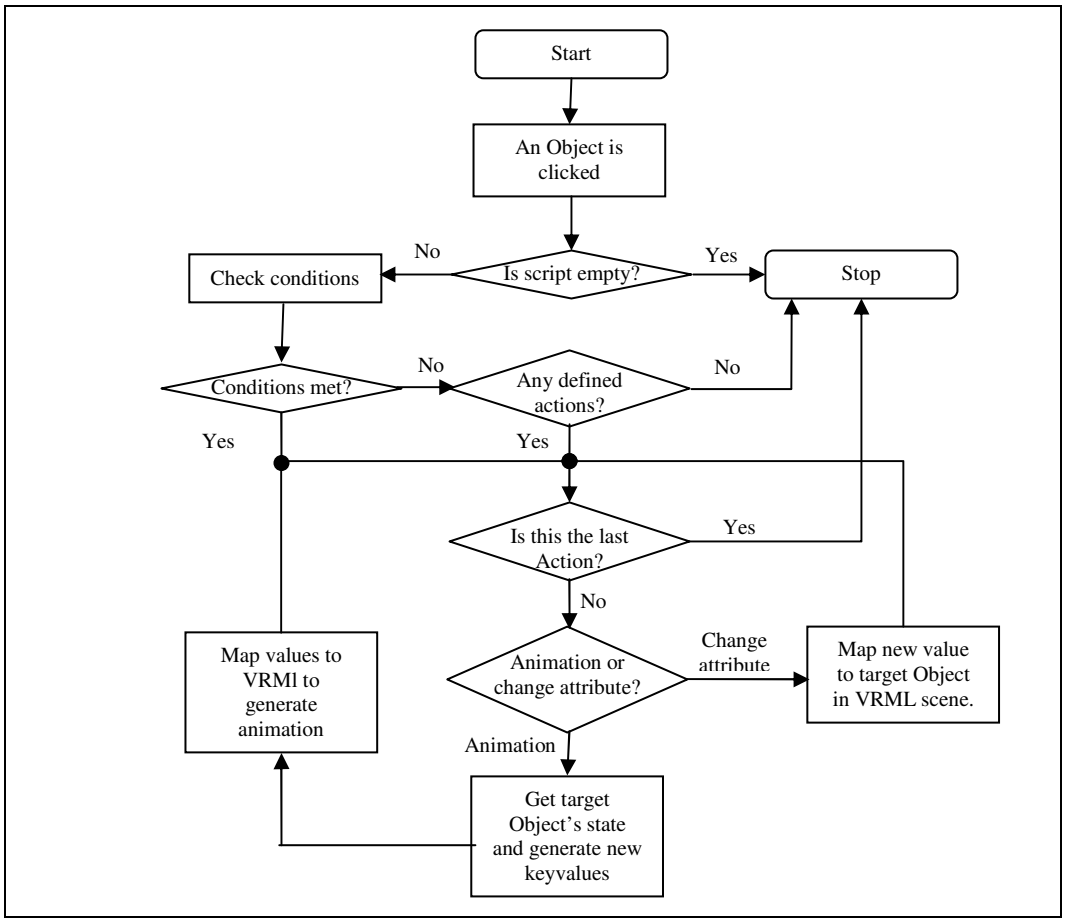


Figure 12 Flow of Events When Object is Clicked

When an Object is clicked, Behavior Controller first checks if the Object being clicked on has an associated Behavior. If Behavior definitions are present, the conditions will be checked. When checking for conditions, the VRML Translator will get the current state of the Object in question from the VRML scene and compare it to the conditions defined. The author may specify actions that take place when conditions are met, or when no conditions are met. The procedure for carrying out these actions is the same.

For an action that involves applying an Animation to a target Object, the VRML Translator first retrieves the Object's latest state, while the Animation

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Controller processes the keyvalues based on the Object's state and the keyvalues of the assigned Animation. Finally, the VRML Translator maps the new keyvalues to the target Object in the VRML scene. For an action that involves changing a target Object's attribute, the VRML Translator simply maps the new values to the target Object in the VRML scene.

4.3.2 CONDITIONS CHECKING

When an Object is clicked, the mouse-click event triggers an event-handling function callback() which implements the operation(s) to be carried out in response to the event. Each Object in the VRML scene has a corresponding event-handling function that determines how the Object reacts in response to mouse-click events. The prototype implements a special method isClicked() for every java object that has a corresponding VRML Object in the scene. Based on the script entered by the author, the isClicked() function processes the conditions and actions and determines the next action to be taken by the Object(s) as a result of the mouse-click event.

Each script definition contains zero or one **IF** block, any number of **ELSEIF** blocks, and zero or one **ELSE** block. Each **IF** and **ELSEIF** block contains one or two conditions, as well as one or more statements. Each **ELSE** block contains no conditions, but may contain one or more statements. Thus, the algorithm for the isClicked() method is derived as shown in Figure 13 using pseudo code.

The algorithm illustrates how processing of the scripts is done. When checking for conditions, the Object's current state is retrieved from the VRML scene and its attribute values (for example position) are compared to that stated in the condition. If the values match, then the condition is met. When processing two conditions, the processing logic is as illustrated in Figure 14.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library


```
If (IFstmt is not empty)
{
       Check for IF conditions
       If (IF conditions are met)
       {
              Process IF statements
       }
       Else
       {
              If (ELSEIFstmt is not empty)
              {
                     Loop through every ELSEIF block
                     {
                            Check for current ELSEIF conditions
                            If (conditions met for current block)
                            {
                                   Process statements for current
                                   block
                            }
                            Else
                            {
                                   Go to next ELSEIF block
                            }
                     }
              }

              If (no conditions met yet)
              {
                     If (ELSEstmt is not empty)
                     {
                            Process ELSE statements
                     }
              }
       }
}
```

Figure 13 Algorithm for isClicked() Method

```
Check if conditions are "AND" or "OR".
AND:
F&&T = F
F&&F = F

T&&T = T
T&&F = F
Final outcome is directly dependent on second condition. Thus, if
the first condition is not met, there is no need to process the
second condition.

OR:
T||T = T
T||F = T

F||T = T
F||F = F
Final outcome is directly dependent on second expression. Thus, if
the first condition is met, there is no need to process the second
condition.
```

Figure 14 Processing Logic for Conditions

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

When processing statements, new values of attributes (for example, color) will be applied to the target Object(s); Animations may also be applied to target Objects. Animation keyvalues are calculated at runtime as these values depend on the attribute values of the Object at the time the Animation was invoked. Thus, the Animation will be smooth and continuous as it starts from the Objects current state.

4.3.3 WORKFLOW FOR GENERATING ANIMATIONS

After checking the conditions, the correct actions have to be invoked. If the action(s) include generation of Animations, the method startAnimation() will be invoked.

Figure 15 shows the pseudo-code for generating Animations. Generation of animations is essentially a two-step process - processing the Animation information, and generation of the Animation. When processing an Animation, the keyvalues of the new Animation are added to the Object's current attribute value. In addition, if more than one Animation is applied to an Object, the keyvalues for the animated attributes will be added to the Object's existing keyvalues. Finally, the processAnimation() method checks for illegal values. For example, the value for Color should lie between 0 and 1. After the necessary keyvalues have been generated, the previewAnimation() method is invoked. This method extracts the necessary Interpolator nodes from the VRML scene before sending EventIns via the changeEAI() method. The changeEAI () method determines the field type of each EventIn before sending in the request to the VRML. This is because the data type to be sent in has to match that of the target VRML node.

```
do_startAnimation
begin
       do_processAnimation;
       do_executeAnimation;
end

do_processAnimation
begin
       do_generateKeyValues;
       do_checkForIllegalValues;
end
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library


```
do_executeAnimation
begin
       do_getInterpolatorNodes;
       do_sendEventInToNodes;
end
```

Figure 15 Viewing Animations

4.3.4 SEQUENCE DIAGRAM

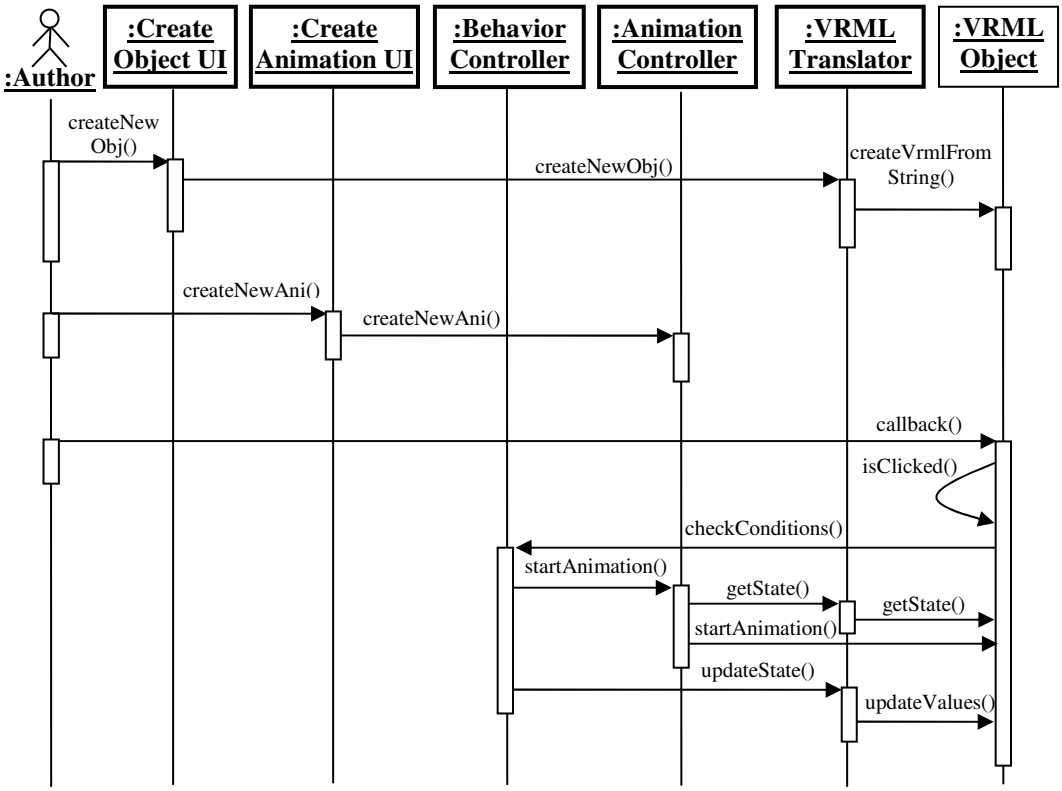


Figure 16 Sequence Diagram

We have identified the major components for implementing the core functionalities. The sequence diagram in Figure 16 shows the sequence of messages sent between the major components when an author creates an Object, an Animation, and when he clicks on an Object in the VRML Scene.

In Figure 16, one possible sequence of messages sent when an author clicks on an Object in the VRML scene is shown. In this sequence, the actions carried out include applying an Animation to an Object, and updating the state of an

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Object. The Behavior Controller module processes the author-defined scripts and calls the Animation Controller when an Animation is to be started. The Animation Controller then calculates the new keyvalues and sends them to the target Object. Similarly, when updating a target Object's state, the VRML translator will be invoked to send a message to the VRML scene.

4.3.5 IMPLEMENTATION – COMPONENTS

The core modules encapsulate the main functionalities of the proposed system. In the implementation model, the core modules are broken down into separate classes as shown in Figure 17. The figure also shows other supporting classes that are implemented in the prototype.

The GUI consists of the mainEditor JApplet interface, as well as several JFrames that pops up when an author wants to add a new Object/Animation, edit an Object/Animation/Script, or edit the Scene Description. An additional class TreeControl is implemented for providing methods to display the data in a JTree interface.

The functionality of the VRML Translator is implemented by two classes – VRMLBuilder, and VRMLObject. The VRMLBuilder provides methods for adding new Objects to the VRML scene, as well as removing existing Objects from the scene. Each 3D Object in the VRML scene has a corresponding VRMLObject instance and an ObjInfo instance. The VRMLObject instance implements the callback() method and allows the Object's latest states to be retrieved and updated.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

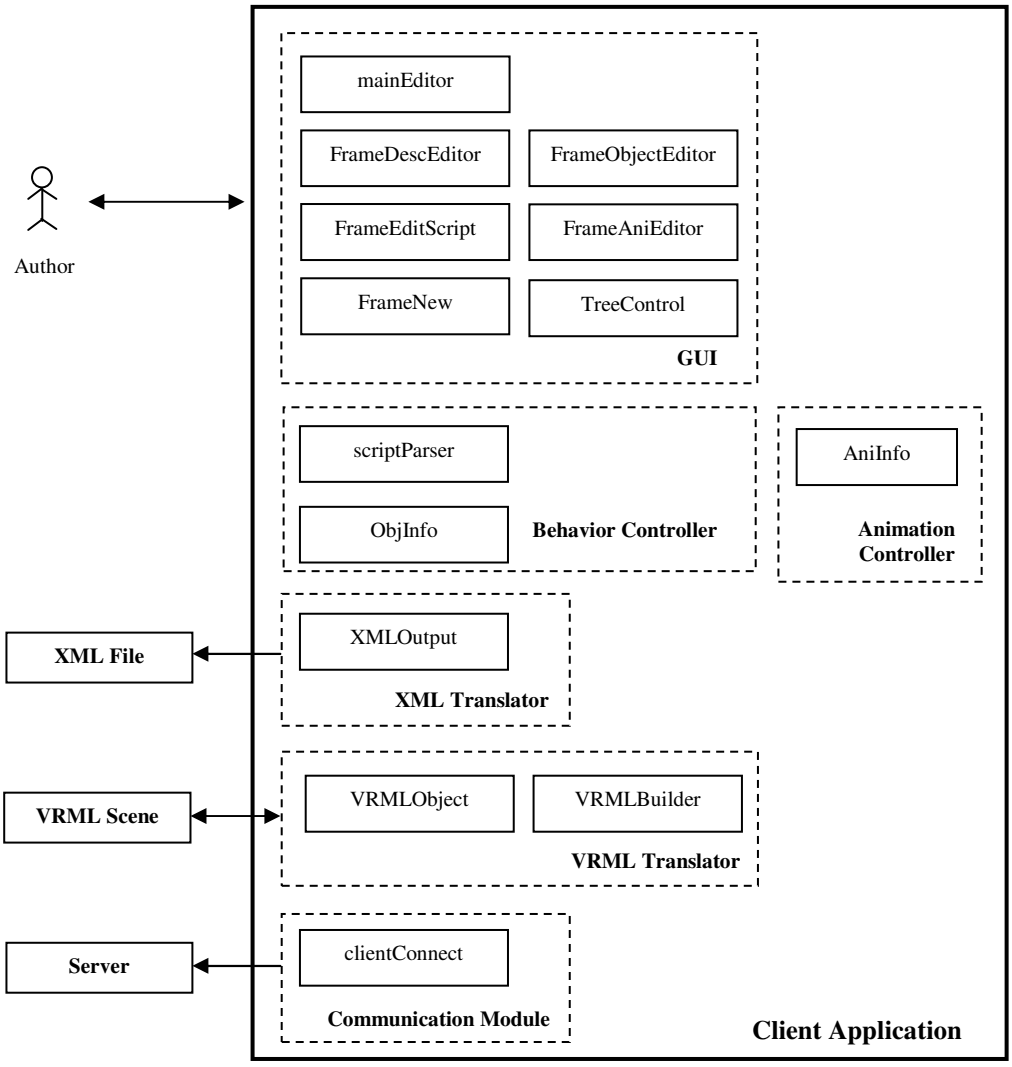


Figure 17 Components of Client Application

The Behavior Controller consists of two classes – scriptParser, and ObjInfo. The scriptParser class provides methods for checking the correctness of the script syntax. The ObjInfo class stores the 3D Object's definitions as well as provides methods for getting the Object's latest state from the VRMLObject instance.

The AniInfo class is instantiated for each Animation defined. it stores the definitions for the Animation, and provides methods for processing the mathematical functions.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

The XMLOutput class parses the XML file containing the Scene Definition at the initial page load, and extracts the necessary information. Before sending the updated information to the server, the XMLOutput also formats the information defined by the author into an XML file and saves it temporarily in the client's local drive before sending it to the server via the clientConnect class. The sole purpose of the clientConnect class is to submit the updated XML file to the server.

All the classes are placed in a single jar file and downloaded onto the client machine for execution.

4.3.6 IMPLEMENTATION – PSEUDO-CODE & SOURCE CODE

In this section, we present the pseudo-code and source code of some of the critical functionality of the prototype.

```
public VRMLObject VRMLGen(ObjInfo objNew) {
       VRMLObject vrmlNew = new VRMLObject(browser, objNew);
       EventInMFNode addChildren = (EventInMFNode) Scene.getEventIn
                                          ("addChildren");
       addChildren.setValue(vrmlNew.Object_array);
       objNew.vrmlObj = vrmlNew;
       return vrmlNew;
}
```

Figure 18 Adding New Object to Scene

Figure 18 shows the souce code for adding a new Object to the VRML scene. The function returns a java object VRMLObject. The definitions entered by the author are first stored in an ObjInfo instance. This information is used to create a new VRML Object in the scene.

Figure 19 shows the source code for updating the VRML scene. In this sample code, we discuss how an Object's position is updated. The new position of the Object is stored in the variable **Pos**. The required EventIn and the target VRML node are then used to update the VRML scene via the changeEAI() method. The changeEAI() method is implemented in the class VRMLObject; it updates Object attributes via standard VRML event calls.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library


```
public void updateChanges() {
       int field_type;
       boolean result;
       String Field;
       String Pos;

       Pos = pos_x + " " + pos_y + " " + pos_z;
       …

       VRMLObject vrmlObj = this.vrmlObj;
       EventInSFVec3f Translation = vrmlObj.Translation;
       …
       vrml.external.Node transform = vrmlObj.transform;
       …
       Field = "translation";
       field_type = vrmlBuild.getFieldType(transform,Field);
       result = vrmlBuild.changeEAI(transform, Field, new
       MFValue(field_type, Pos));
}
```

Figure 19 Updating VRML Scene

```
public void processFunctions() {
       …
       for (i = 0; i < n; i++)
       {
         //allow variable t in function
         jep.addVariable("t", i);
         //parse the expression
         jep.parseExpression(str_Defn);
         attribute[i] = (float)jep.getValue();
       }
       …
}
```

Figure 20 Generating Animation Keyvalues

Figure 20 shows the sample code for generating keyvalues for an Animation. The variable **n** represents the number of keyvalues to generate. Generally, a larger number of keyvalues results in a smoother animation path. However, generating more keyvalues also mean more computing resources is required; in the prototype, **n** is set to a default value of 6. The **jep** variable is an instance of the JEP mathematical functions parser class. It contains methods for parsing mathematical functions.

The expression defined by the author (**str_Defn**) for that particular attribute is parsed by the parseExpression() method. The return value is then added to an array that stores the keyvalues of this attribute (for example, position).

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

The code sample in Figure 21 shows how Animation is processed. The sample shows how the keyvalues for animating an Object's position is updated

KV_Pos_x[], **KV_Pos_y[]**, and **KV_Pos_z[]** arrays are used to store the keyvalues for animating the position of the Object. The function resetValues() resets the attributes to their default values; for position, the default is x = 0, y = 0, and z = 0. After this is done, the keyvalues are updated by adding the Animation keyvalues to the Object's current position values. The checkValues() function checks for illegal values in the keyvalues. For example, when specifying an Object's color, the range of values should be between 0 and 1.

```
public void processAnimation(AniInfo curAni, int n) {
       resetValues();
       //calculate new values
       for (int j=0; j < n; j++)
       {
           KV_Pos_x[j] = pos_x + curAni.KV_Pos_x[j];
           KV_Pos_y[j] = pos_y + curAni.KV_Pos_y[j];
           KV_Pos_z[j] = pos_z + curAni.KV_Pos_z[j];
           …
   }
   checkValues();
       KFPos = String.valueOf(KV_Pos_x[0]) + " " +
              String.valueOf(KV_Pos_y[0]) + " " +
              String.valueOf(KV_Pos_z[0]) + ", " +
              …
              … ;
}
```

Figure 21 Code Sample for Processing Animations

4.4 THE AUTHOR INTERFACE

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

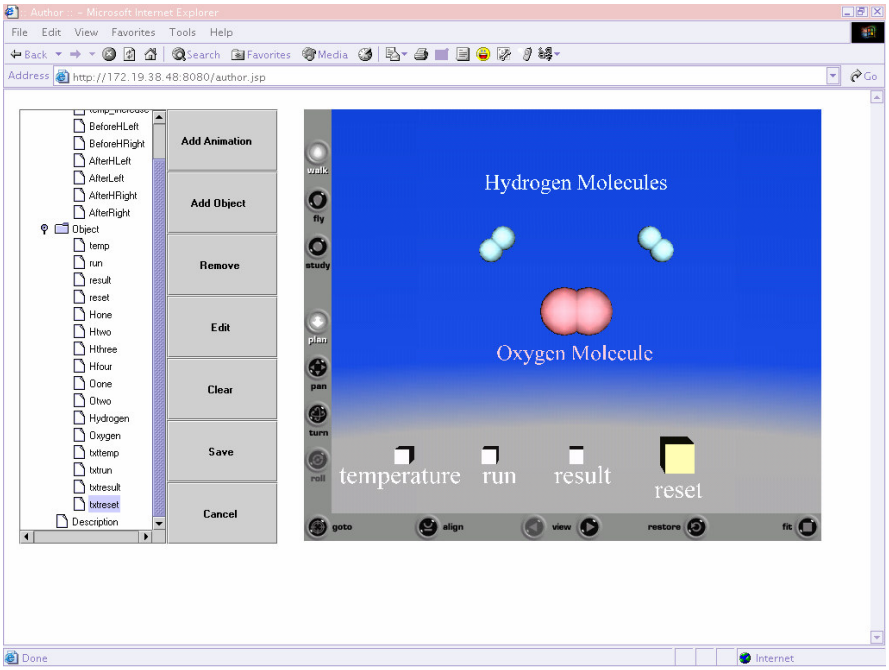


Figure 22 Overview of Author's webpage

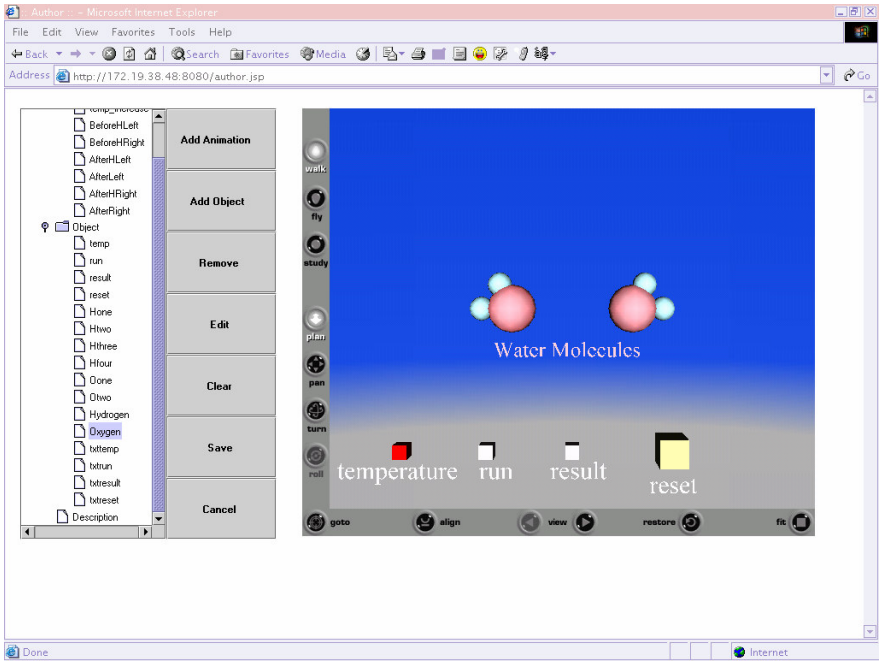


Figure 23 Playing Animation in Author's Window

Figure 22 and Figure 23 show the control panel for the author. The tree structure on the left shows the list of Animations, Objects, and Descriptions defined. Using the various control buttons on the control panel, the author is able to create new Objects and Animations, remove existing ones, as well as edit their properties. The author is also able to clear the entire scene and start again. Clicking on the "Save" button periodically ensures that the scene is

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

saved on the web-server and can be recovered in the event of browser crash or when network problems occur.

The author is also able to preview animations in the authoring interface. Figure 23 shows the result of playing the animation that shows how water molecules are being formed from oxygen and hydrogen molecules. Figure 24 shows the Script Editor Window in which an author used to enter the scripts for controlling an Object's behavior. Figure 25 shows the Scene Description Editor Window. The author is able to enter the Scene Description in the text area.

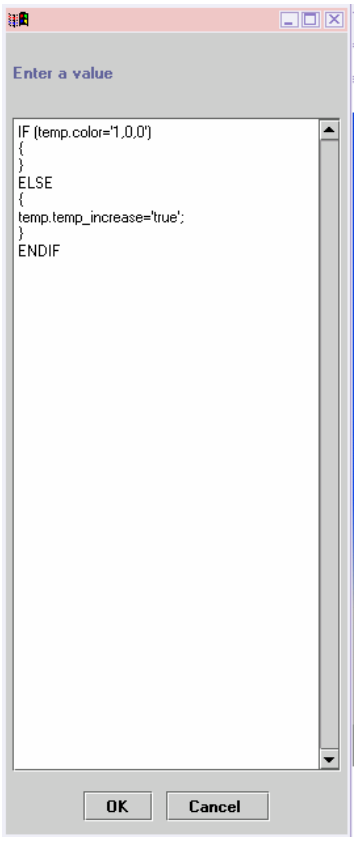


Figure 24 Script Editor Window

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

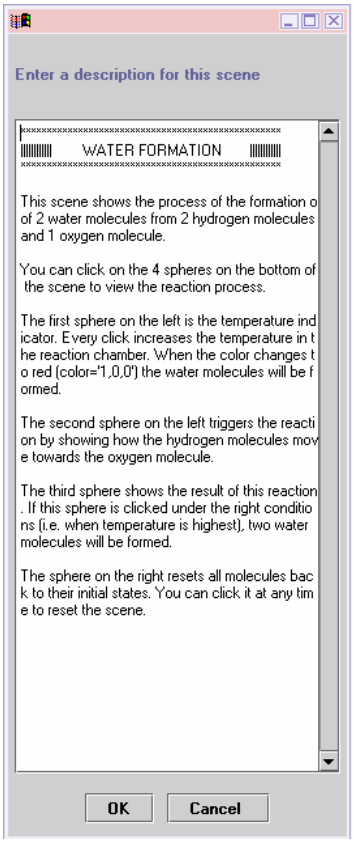


Figure 25 Scene Description Editor Window

Before proceeding to the reader interface, we will present a simple case study on creating an event-triggered animation.

4.4.1 CASE STUDY OVERVIEW

In this case study, the author is interested in creating a visualization of the control system described in Table 2. An Object named BALL is controlled by two external conditions – temperature, and switch position.

Table 2 States and Outcomes Table

Condition		Outcome
Temperature	**Switch Position**	
Don't Care	0	Stationary
LOW	1	BALL moves at constant speed in x-direction

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

HIGH	1	BALL moves at constant speed in x-direction while oscillating along y-axis
LOW	2	BALL oscillates along z-axis
HIGH	2	BALL oscillates along x-axis

In this case study, the author wants to show the reader how the motion of an Object (BALL) is affected by the two external conditions. The behavior rules for this case study are created for the purpose of illustrating how the web-based 3D visualization environment can be used.

We shall study how the author can create a meaningful visualization using the authoring tool.

4.4.2 DEFINING CONTROL MECHANISMS

The author needs to provide mechanisms for the reader to control the two conditions (temperature and switch position). These can be in the form of push buttons, which a reader can click on. It would also be useful if the reader can tell the state of the environment from these push buttons. Thus, the author may indicate the temperature by setting the color of the temperature button to red when it's hot and setting it to blue when it's cool. The script for achieving this effect is shown in Figure 26.

When the object called temp_control is clicked on, the script is triggered. The script shown in Figure 26 simply toggles the color of the temp_control object to indicate whether it is 'hot' or 'cold'.

```
IF(temp_control.color==’1,0,0’)
{
       temp_control.color=’0,0,1’;
}
ELSEIF(temp_control.color==’0,0,1’)
{
       temp_control.color=’1,0,0’;
}
ENDIF
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Figure 26 Script for Temperature Button

For the switch control button, a reader may find it helpful if the current state is displayed prominently below the switch button. Thus, the author may create a text object below the switch control button to indicate the current switch position. The reader may change this value anytime by clicking on the switch control button (Object named switch_control), and the new value will be updated instantly. The script for updating the text object is shown in Figure 27. This script should be defined under the Object switch_control; it will then be run each time Object switch_control is clicked on, and the value of switch_display will be updated accordingly.

```
IF(switch_display.text=='0')
{
       switch_display.text='1';
}
ELSEIF(switch_display.text=='1')
{
       switch_display.text='2';
}
ELSEIF(switch_display.text=='2')
{
       switch_display.text='0';
}
ENDIF
```

Figure 27 Script for Switch Control Button

At this point, the following Objects have been identified and defined.

Table 3 Control Objects Defined

Object name	Description
temp_control	Spherical Object. Script allows reader to change its color at each click. Color indicates temperature and can be used to control other Animations.
switch_control	Spherical Object. Script allows reader to change the text displayed in text Object [Switch_Display] at each click. The text displayed indicates the switch position and can be used to control other Animations.
switch_display	Text Object. This Object displays the switch value.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

The author may want to create another control mechanism that resets the system to a default condition. The script for the Object named "Reset", used to reset the system, may be defined as shown in Figure 28.

As the condition `reset.pos = '3,3,3'` is always met, clicking on the reset object will always result in the actions defined.

```
IF(reset.pos=='3,3,3')
{
        switch_display.text='0';
        temp_control.color='0,0,1';
        BALL.pos='0,0,0';
}
ENDIF
```

Figure 28 Script for Reset Button

With all the control mechanisms ready, the author may like to think of how he should arrange the Objects in 3D space. For this case study, the Objects are arranged as shown in Figure 29. Additional text Objects are created to label each control button; this allows the reader to know the function of clicking on each button.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

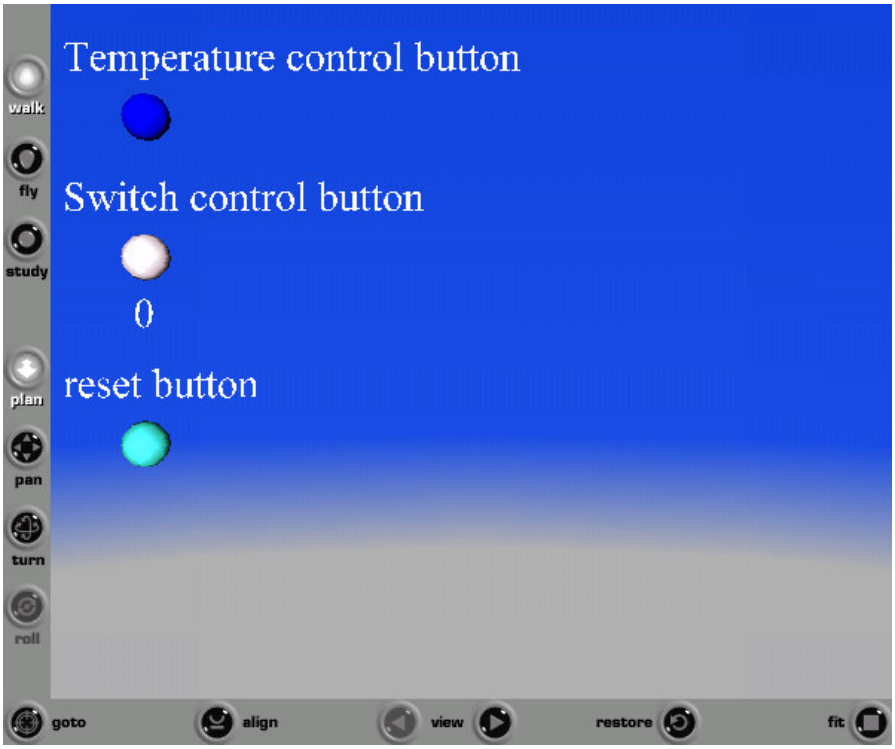


Figure 29 Control Objects

4.4.3 DEFINING THE ANIMATIONS

Before the author proceeds to create the Object BALL, whose motion is to be studied by the reader, he should define the list of possible Animations exhibited by the Object BALL under different conditions.

The paths defined in Table 4 can be described using the following equations.

Table 4 Equations that Describe the Paths of BALL

Animation Name	Description	Mathematical Equation
Con_x	Constant speed in x-direction	Pos_x = 5*t
Osci_x	Oscillate along x-axis	Pos_x = 5*sin(3.142*t)
Osci_y	Oscillate along y-axis	Pos_y = 5*sin(3.142*t)

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Osci_z	Oscillate along z-axis	Pos_z = 5*sin(3.142*t)

4.4.4 SCRIPTING THE BEHAVIORS

In this visualization, the reader is able to view the path of motion of the Object BALL under different conditions by clicking on the Object BALL. Table 5 shows the two conditions (temperature, switch position) that control the path of the BALL. Based on the relationships between the color of temp_control and value of switch_display, a new conditions table can be drawn.

Table 5 Conditions Table

Conditions		**Outcome**
Temp_control.color	**Switch_display.text**	
N.A.	0	nothing
0,0,1	1	BALL.con_x='true'
1,0,0	1	BALL.con_x='true' BALL.osci_y='true'
0,0,1	2	BALL.osci _z='true'
1,0,0	2	BALL.osci _x='true'

From Table 5, we then derive the script used to control BALL's path of motion in a straightforward manner.

Figure 30 shows the script that is run when a user clicks on the object BALL. Based on the script, after the object BALL is clicked on, the color of the object temp_control and the value of the object switch_display will be checked. Different actions will be invoked depending on the states of the objects temp_control and switch_display. For example, when the color of the temp_control is blue and the switch is at position 2, the BALL will undergo the animation called osci_z.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library


```
IF(switch_display.text=='0')
{

}
ELSEIF
(temp_control.color=='0,0,1' && switch_display.text=='1')
{
BALL.cons_x='true';
}
ELSEIF
(temp_control.color=='1,0,0' && switch_display.text=='1')
{
BALL.cons_x='true';
BALL.osci_y='true';
}
ELSEIF
(temp_control.color=='0,0,1' && switch_display.text=='2')
{
BALL.osci_z='true';
}
ELSEIF
(temp_control.color=='1,0,0' && switch_display.text=='2')
{
BALL.osci_x='true';
}
ENDIF
```

Figure 30 Script for BALL

4.4.5 DISCUSSION ON CASE STUDY

This example thus illustrates the following:

- Strengths of the tool:
    - Animation Definition – animations could be defined in a single equation. For animation paths which are governed by known formulae, this is a useful feature as it eliminates the need for authors to calculate and define keyvalues to generate animation frames.
    - Behavior Scripting – learning from Alice's simple scripting language, our proposed language uses a basic syntax construct `<Object>.<animation or attribute> = <value>`. An informal survey with non-CS students revealed that this construct is simple to understand. Furthermore, it could be extended in future to include different attributes of an object; this will allow a larger variety of animations to be created. For example, in addition to changing basic attributes like position, orientation, and color, one could change the texture and shape of an object over time.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

- Code Reusability – the author can define an animation once and apply it to different objects. For example, to apply the animation to the object BALL, the author can type this statement `BALL.osci_x='true'`. To apply the same animation to another object called WHEEL, the author can type the statement `BALL.osci_x='true'`.

4.5 THE READER INTERFACE

The components that are used in the client application for the author can be re-used in the client application for the reader. The reader is able to view the VRML scene created by the author, as well as the Description of the scene written by the author. However, as readers do not edit the scene information, the reader's application will have a different GUI that does not provide access to scene authoring. Instead, the GUI only has one button "View Details" as shown in Figure 31. This button allows a reader to see the details of an Object or Animation.

A list of defined Objects and Animations displayed allows the reader to select any of them to view more information. This allows a reader to view the current state of the selected Object. The scripts that define the Behaviors of Objects when this Object is clicked will also be displayed, as shown in Figure 32. In Figure 32, the script for Object BALL is displayed; it shows how different objects in the scene will behave based on the different conditions.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

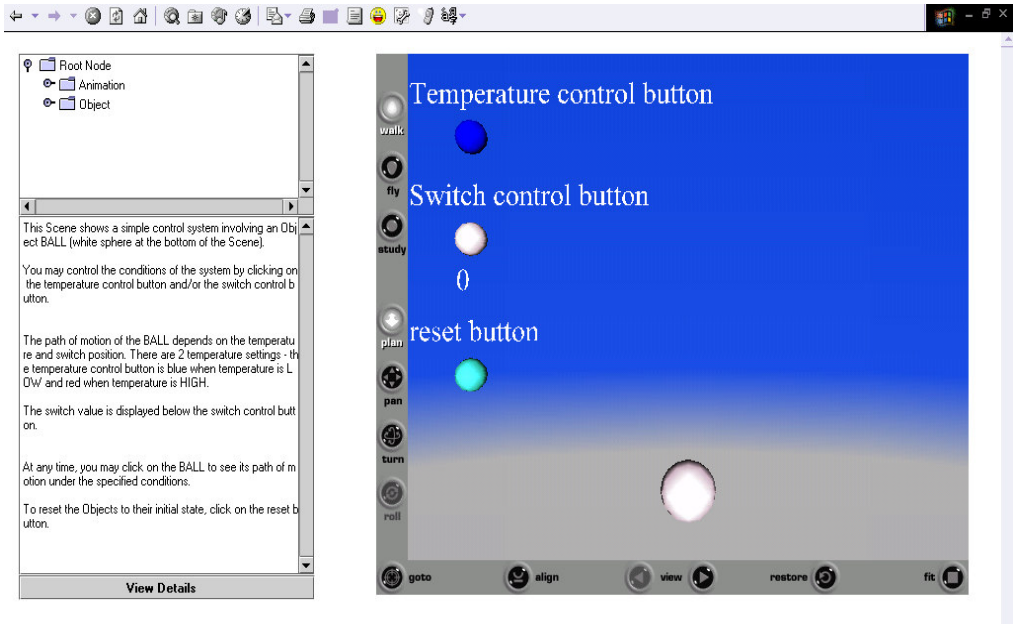


Figure 31 Overview of Reader's Webpage

More importantly, the 3D Objects in the visualization will behave as defined by the author. This allows the reader to view the 3D Objects, as well as watch their behaviors in response to external events such as mouseclicks.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

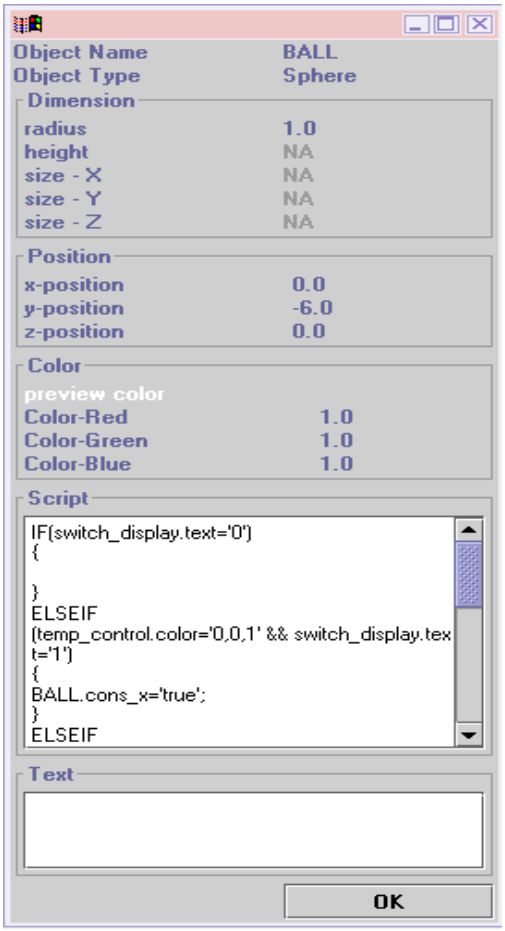


Figure 32 Viewing an Object's Details

4.6 SAMPLE APPLICATIONS AND DISCUSSIONS

In this section, we give an overview of the website that was designed to support the web-based 3D authoring tool. It is a sample application that makes use of the proposed web-based 3D authoring tool. In addition, we also discuss the main advantages of our proposal based on the proof-of-concept prototype.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

BASIC WORKFLOW

The basic workflow of the sample application is shown in Figure 33 below.

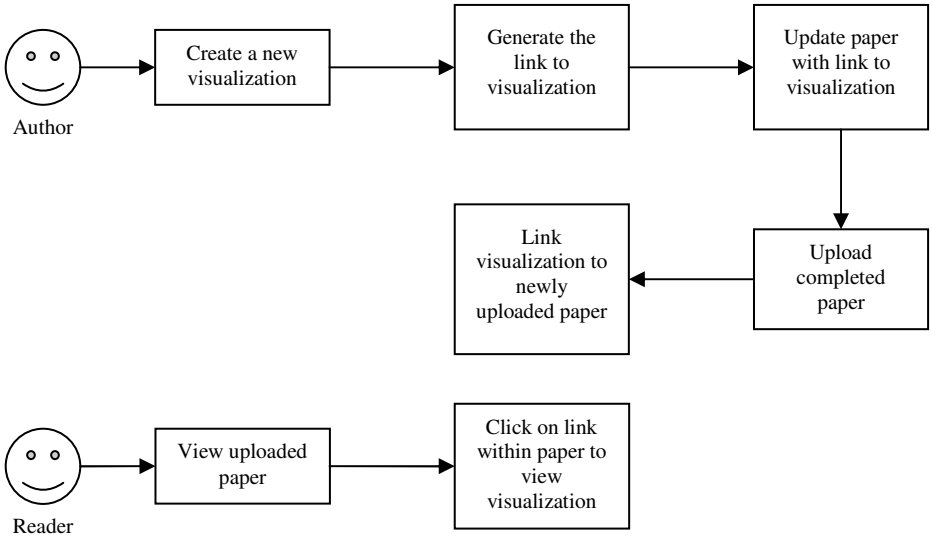


Figure 33 Basic Workflow of Sample Application

The author first creates a new visualization using the online tool provided. After the new visualization is saved, a link to the visualization will be generated. The author can then copy this link and paste it into the paper he is going to upload to the system. Doing so allows the reader of the paper to click on the link and access the visualization.

The following section will discuss the sample application in greater detail.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

SAMPLE APPLICATION

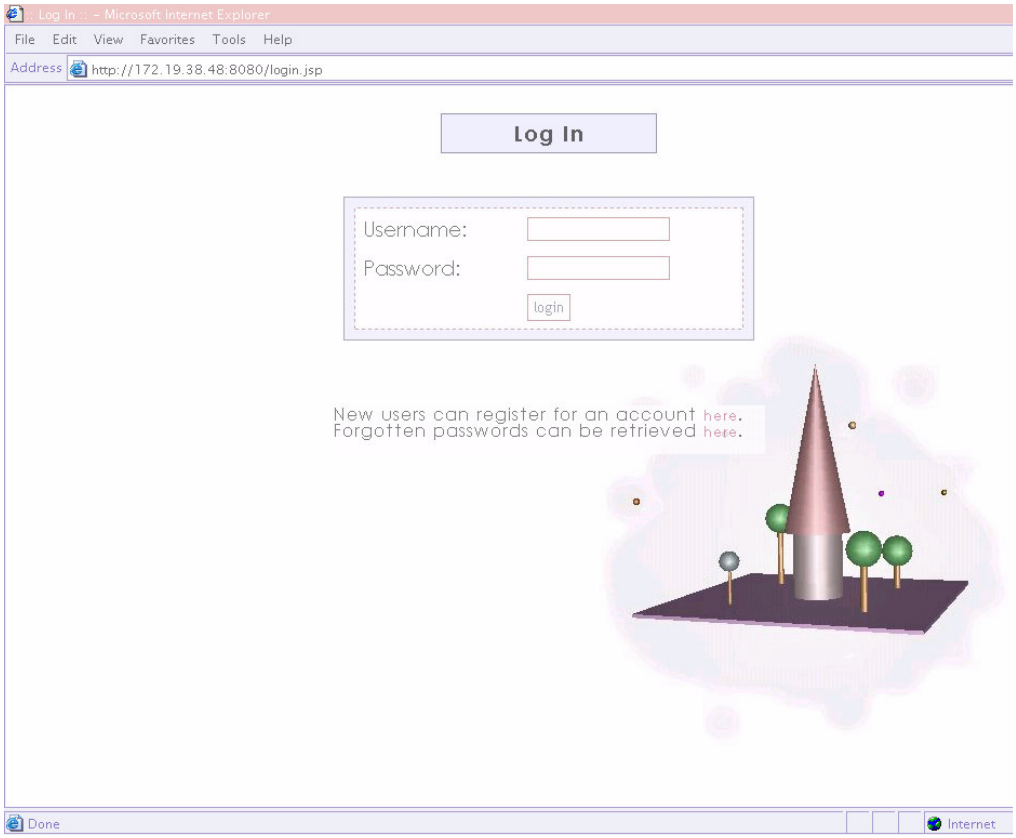


Figure 34 Author Login Screen

Figure 34 shows the login screen for authors. The prototype provides mechanisms for registering new users to the authoring system, as well as password retrieval for existing users. An author can upload new documents, as well as create, edit, preview, and delete 3D visualizations after logging in to the system.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

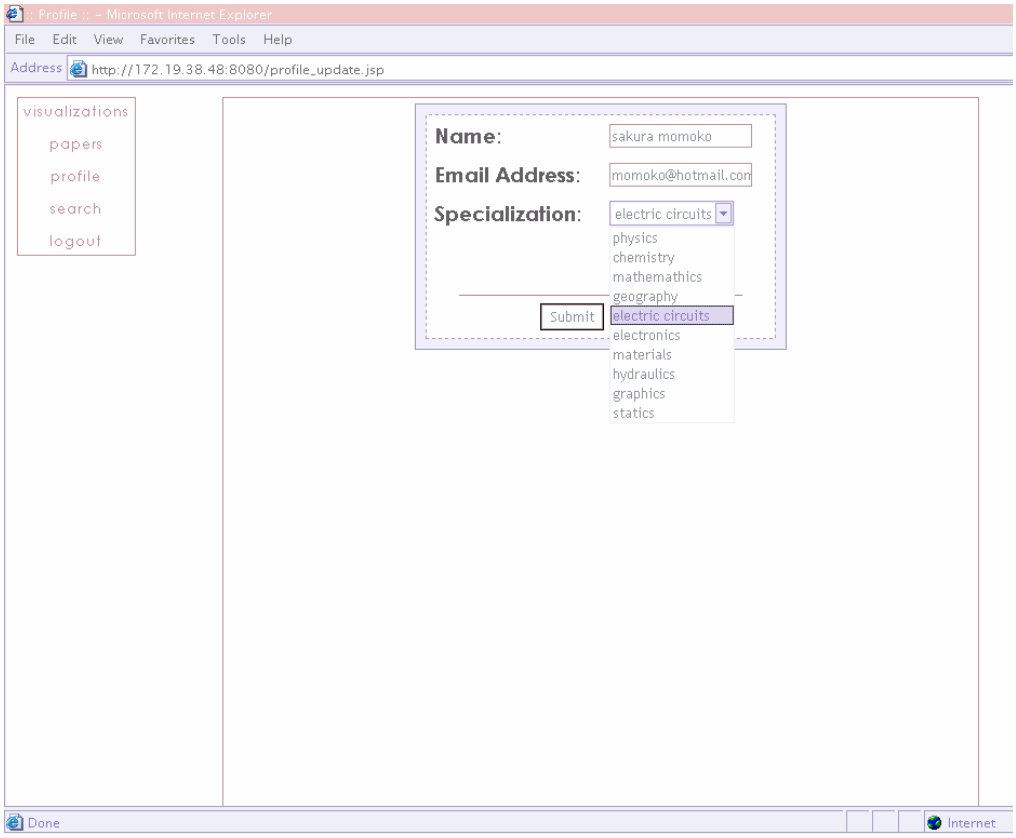


Figure 35 Update Profile Screen

Figure 35 shows the Update Profile Screen for the author. An author is able to update his full name, email address, and area of specialization. In the event that the author is unable to log in to the system because he has forgotten the password, the system will send the password to this email account specified by him.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

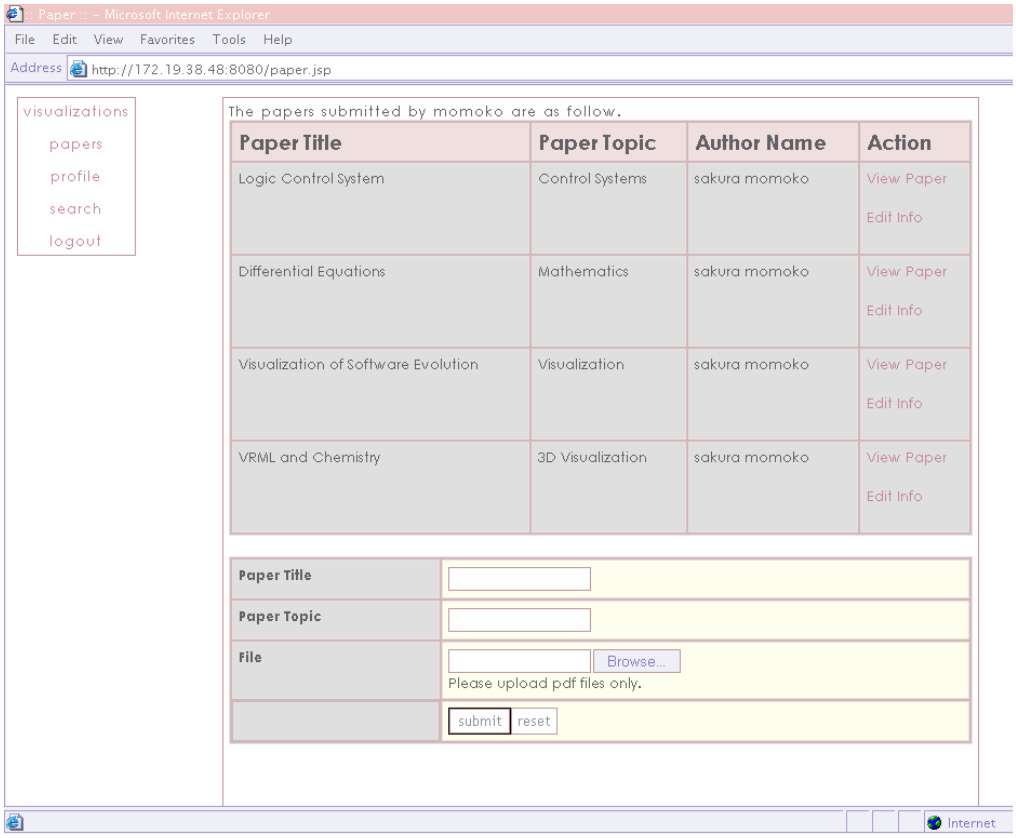


Figure 36 View Uploaded Papers Screen

The sample application allows authors to upload papers onto the management system, as well view the 3D visualizations that are linked to these papers. Readers will be able to view the papers uploaded by authors and click on links to view the 3D visualizations.

Figure 36 shows the View Uploaded Papers Screen. An author is able to view a list of papers uploaded previously. Links are provided to allow the author to view the uploaded papers, as well as edit information on the uploaded papers. A form is also provided for allowing the author to upload new papers to the system.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

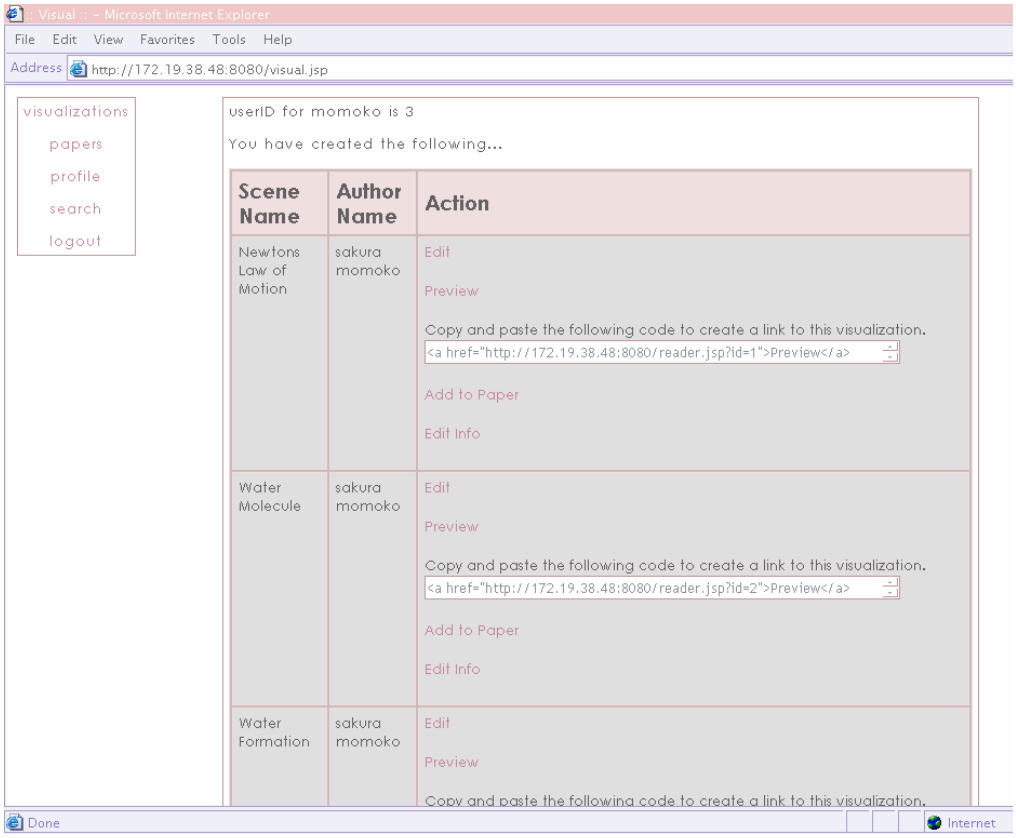


Figure 37 View Visualizations Screen

Figure 37 shows the View Visualization Screen; it displays a list of visualizations created by the author. The "Edit" link brings the author to the authoring page as shown in Figure 22; the "Preview" link brings the author to the reader page as shown in Figure 31. To insert a link that links to the visualization in a document, the author can copy the text in the little textarea and paste it into the destination document. As the system recognizes visualizations by IDs (transparent to authors), the author is able to assign the same name to two different visualizations.

The author is able to create a relationship between visualizations and uploaded papers by clicking on the "Add to Paper" links; this is shown in Figure 38 below. The "Edit Info" link allows an author to change the name of the visualization; this is shown in Figure 39 below.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

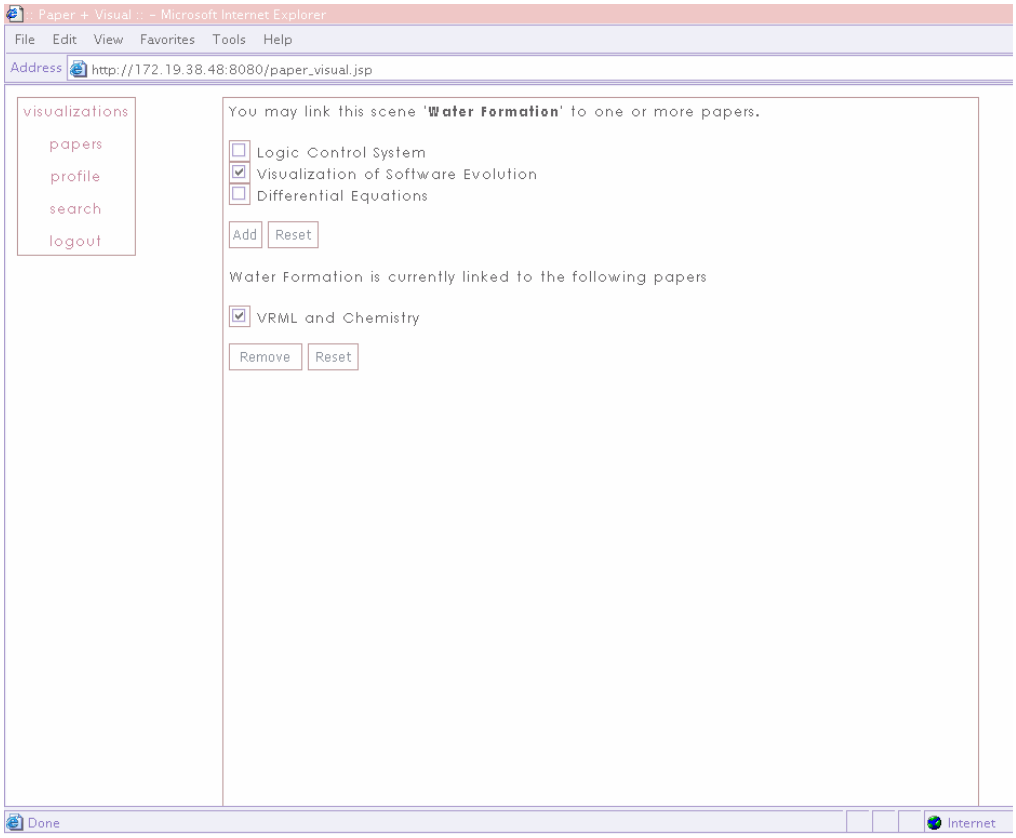


Figure 38 Add Paper to Visualization Screen

Figure 38 shows the Add Paper to Visualization Screen. In this screen, an author is able to assign uploaded papers to the particular screen. He is also able to remove assigned papers from the visualization. This function provides a way for the author to view the relationship(s) between his submitted papers, and visualizations.

As the author is able to edit the visualizations at any time, the uploaded papers will always be linked to the most recently saved versions of visualizations. This eliminates the need for authors to update the links to modified visualizations.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

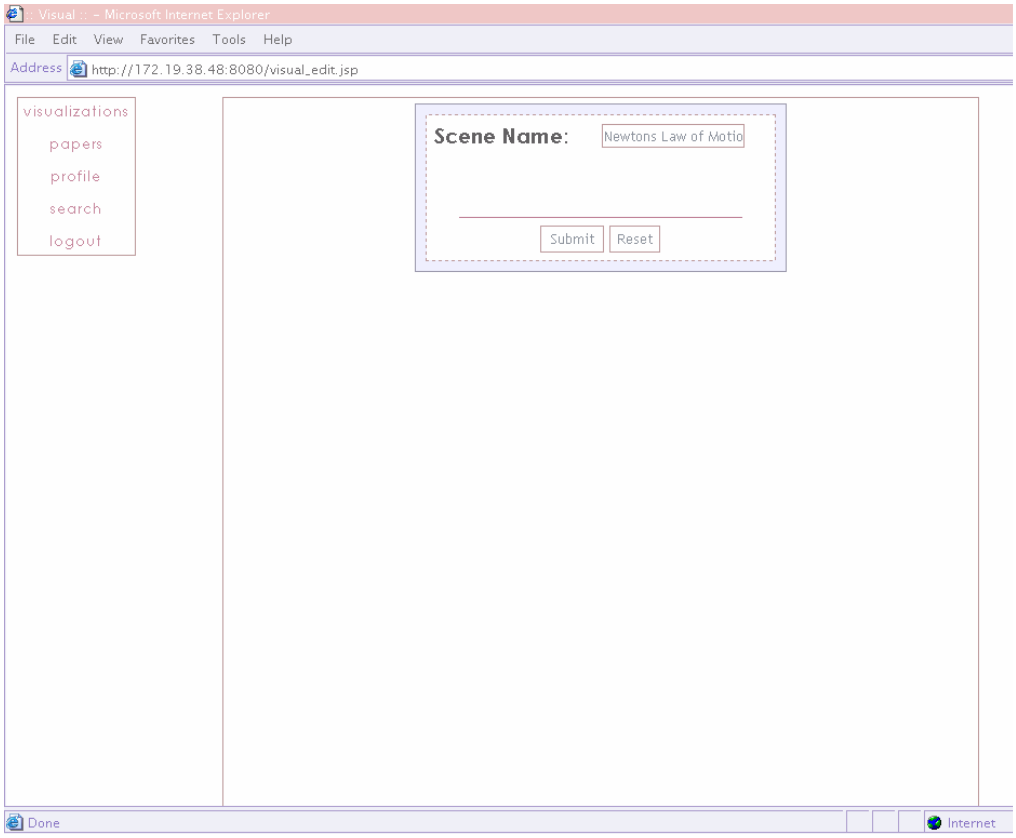


Figure 39 Update Scene Name Screen

Figure 39 shows the Update Scene Name Screen. In this screen, an author is able to change the name of the visualization scene. It is advisable to give a meaningful and descriptive name to the visualization.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

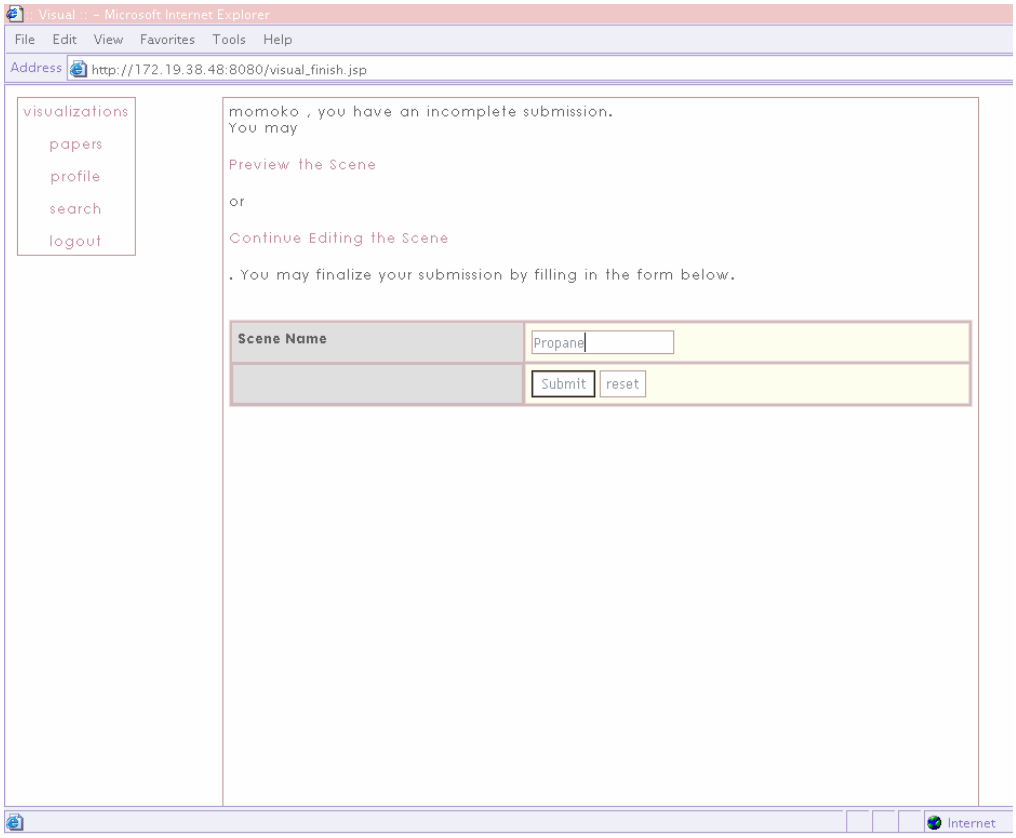


Figure 40 Finalize a Visualization Submission Screen

Figure 40 shows the screen that allows an author to finalize a visualization submission. A new Visualization is considered as "submitted" only after its name has been defined here. An author can continue editing the visualization scene by clicking on "Continue Editing the Scene" if he has a last-minute change of mind at this stage.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

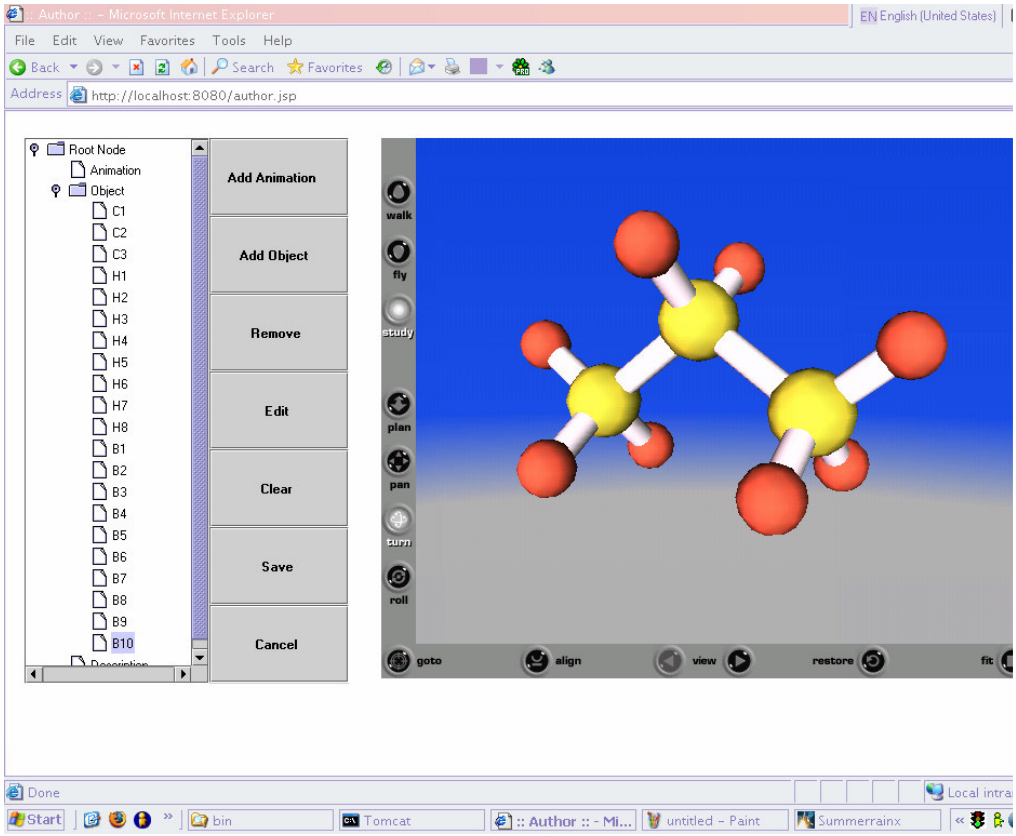


Figure 41 Propane Molecule

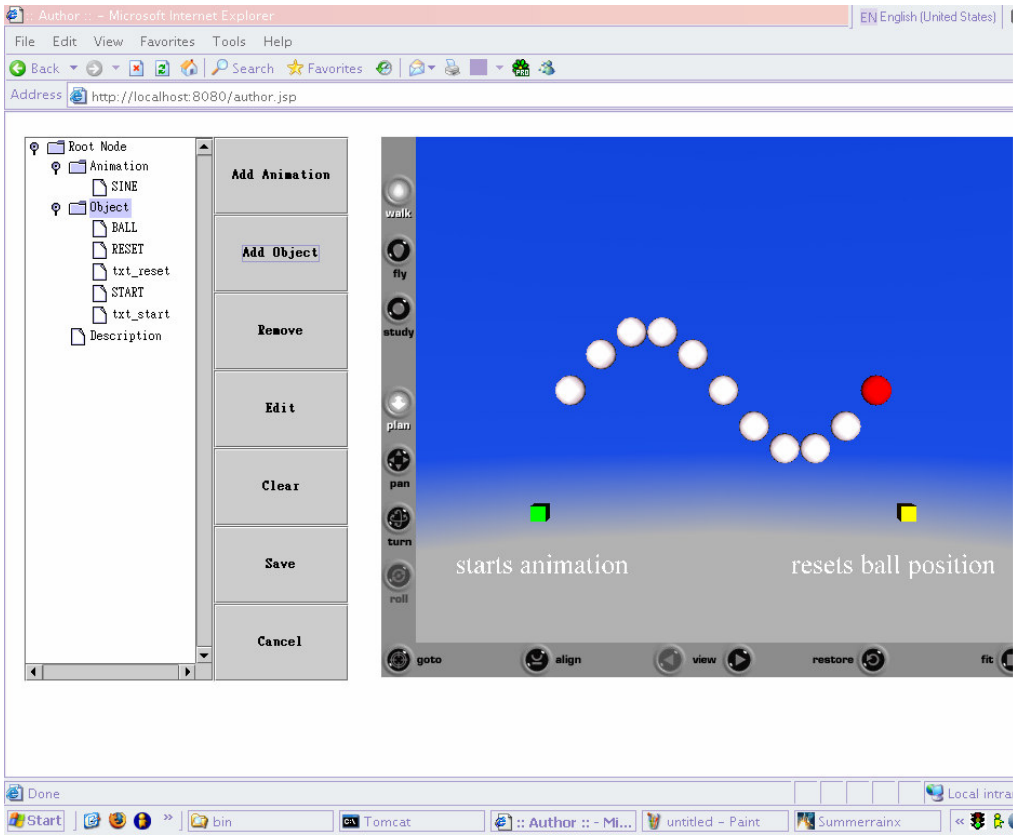


Figure 42 Path of Sine Wave

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Figure 41 shows a screenshot of the prototype. Using the GUI, the author creates a model for Propane molecule (C_3H_8) by combining spheres and cylinders. Figure 42 shows how the author/reader may preview the path of an animation. Animation preview allows an author to check that the path of the animated Object is correct. In Figure 42, the final position of the BALL is denoted by the position of the red ball. The different spheres created in this example allow a reader to visualize the path of the animation.

There are two ways in which an author can set the position of an object. The author can define the x, y, and z positions of an object by clicking on the "Edit" button. Callback mechanisms link VRML objects in the scene to their respective nodes in the java tree structure; by clicking on an object of interest in the VRML scene, its respective node will be highlighted. When the "Edit" button is clicked on, the "Edit Object Properties" window (Figure 43) is displayed, allowing the author to edit the object attributes.

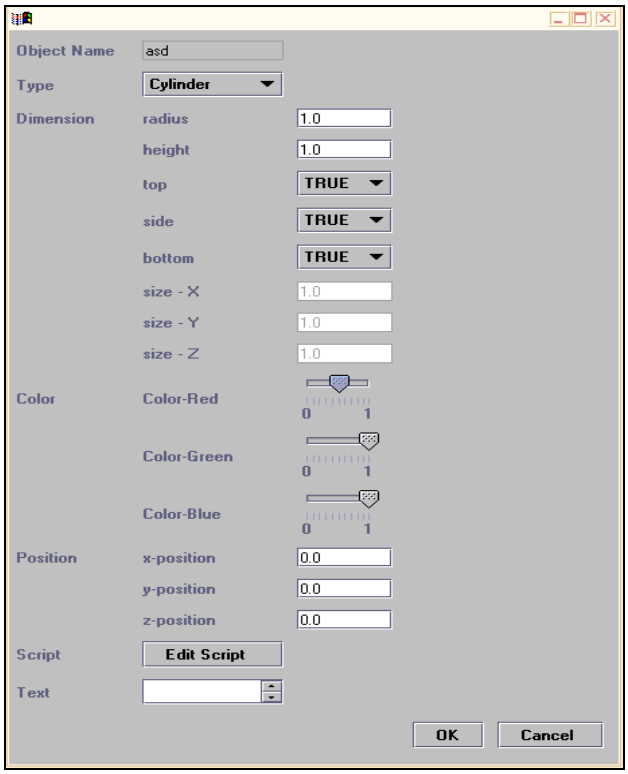


Figure 43 Object Properties Window

Alternatively, when an author drags an object in the VRML scene, causing it to change its position, callback mechanisms will also update the object's x, y, and

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

z attributes accordingly. This ensures that when the author invokes the "Object Properties Window", the attribute values will always be the most updated.

Table 6 shows how an author defines the path of the ball shown in Figure 42 using our proposed mathematical functions-based modeling and the traditional keyframe animation method.

Table 6 Defining a Sine Wave

Using mathematical functions-based method
x = 0.5 * t – 5 y = sin (0.5 * t)
Using normal keyframe animation method
The author has to explicitly calculate the x and y positions for the ball at different instants, before proceeding to define these values at each time instant.

Even though Flash comes with a "tweening" feature that fills in the animation between the first and last frame, the animation path would be more accurate if it is defined frame by frame. Our proposed system allows the author to input formulae for changes in attribute values and generates the values for each keyframe automatically. When using a tool such as Flash, there is a need to calculate and define these attribute values at each time instant and input them into the system. Clearly, our proposed method provides a more convenient way of generating animation paths that are dependent on mathematical rules.

DISCUSSION

Without using complex code and expensive software, authors can create 3D models via a web browser and java applet. This achieves the objective of creating an interface to VRML that allows authors to create visualization for abstract concepts such as chemical models. With the use of our proposed script and Animations, the author may create more interactive models that respond to user mouseclicks. This enhances the value of the application.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

The main advantages of the prototype are as follow:

- Web-based and standards compliant

  As the prototype is web-based and standards-compliant, readers and authors are able to access it via a standard browser with the VRML plugins installed. Thus this system is an excellent tool for sharing of knowledge as any user (author or reader) is able to access it with a standard web browser. Compared to standalone applications, this web-based application reaches out to a larger audience.

- Scripts

  The use of scripts greatly enhances the capability of the web-based 3D visualization system in creating more meaningful visualizations. With the scripts, an author is able to set conditions and resulting actions of certain events. The simple example in our case study illustrates how this can be done.

  An author would have to spend a lot more time creating the same visualization using pure VRML coding. Our prototype web-based 3D visualization system simplifies the author's job by providing a well-structured script syntax that allows the author to create more complex behaviors without going through the complexities of scripting in VRML. In addition, the author does not need to learn about ROUTEs, Interpolators, and Events when using the proposed authoring tool, as opposed to coding in VRML.

- Interactivity

  The prototype allows the reader to interact with the Scene via various Objects which can be clicked on. The reader is able to change the conditions and view the associated outcome, as well as study the states of each Object at any time. Providing details of Animations and Objects to the reader helps the reader to gain a better understanding of the Scene. In addition, the

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

visualization allows the reader to participate by changing the conditions; it is thus more than a simple playback animation.

Therefore, using the proposed framework, we are able to achieve our main research objective of providing an interface to VRML that allows content authors to create animated 3D visualization with well-defined behaviors. The use of EAI enables content authors to create 3D visualizations in VRML with greater ease, while the use of our proposed script enhances the value of VRML by providing an easy way to define behaviors. In addition, the animations are controlled by mathematical-based functions; this allows more accurate modeling to be created while saving content authors the hassle of calculating values of animated attributes at each time frame.

The prototype provides mechanisms to communicate with the VRML scene via standard nodes. With the support of the PROTO node, we will be able to use the same mechanisms to control user-defined nodes and fields. Thus, the variety of Animations that can be generated is greatly increased.

The major drawback of this design is the speed of processing. The prototype suffers from lags in the initial page loading. This is due to the initialization of java applet, parsing of XML file, as well as generation of VRML objects. Using a 1GHz processor on a LAN network, the downloading, processing and generation of VRML scene took between 5-10 seconds each time. With the faster processors available today, the speed of processing is expected to improve.

Additionally, due to the focus on animation and behavior modeling, the sample objects created using this prototype are simple shapes. In future implementations, it is possible to allow users to upload more advanced VRML objects into the scene. Animations and behaviors can then be applied to these VRML objects as a whole, using the concept of motion mapping.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Chapter 5 Conclusion and Recommendations

5.1 CONCLUSION

In this thesis, we described the research work that was carried out in the area of web-based 3D visualization. A novel mechanism that allows authors of electronic documents to create interactive and animated 3D objects was proposed. This mechanism can be used to create visualizations for various ideas or concepts that would be more easily understood with proper visualization.

Our contributions are as follow:

- A web-based 3D visualization environment to allow effective illustration of abstract concepts.
- User-friendly authoring environment for creating dynamic, interactive visualizations.
- Small scripting language for controlling dependencies between the behaviors of 3D objects.
- Mathematical functions-based modeling of object behaviors.

The main advantages of the proposed object modeling mechanism are discussed in greater detail below:

- Firstly, the proposed system is web-based and conforms to web standards. It is thus usable on any web browser with the necessary VRML plugins installed. This allows authors to share their knowledge with anyone who has access to the Internet.
- Secondly, research effort was focused on proposing a mechanism that allows even authors with little or no programming experience to create meaningful visualizations. This allows the proposed system to reach out to a larger group of authors.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

- Thirdly, our proposed mechanism supports mathematical functions-based modeling for animations as opposed to keyframe animation. This mechanism results in a more accurate modeling of animations as the changing attribute values are calculated based on formulae. Furthermore, this mechanism also eliminates the need for authors to calculate and enter keyvalues individually; new keyvalues will be calculated dynamically when changes are made to the mathematical functions.

- The proposed scripting language supports the Event-Condition-Action rule that makes visualization more meaningful. By providing a conditions-checking mechanism, we allow authors to create dependencies among objects. Events and ROUTE statements in VRML cannot achieve this, thus we enhance the visualization by providing mechanisms to control Object behaviors and behavior rules. Furthermore, details and complexities of communicating with the VRML scene are hidden from the authors; authors can therefore focus on telling the system "what to do" instead of thinking about "how to do".

The main disadvantages of the proposed system are as follow:

- To fully utilize the strength of the proposed system in creating objects with inter-dependent behaviors, authors have to learn our small scripting language.

- As the scripting language was designed for the purpose of defining behaviors, it is not a full-fledged programming language. Thus, it may be seen as "limited" due to the lack of features like additional control statements.

- As most of the processing is done on the client machine, some lag may be experienced on machines with slower processors. However, our performance evaluations were done on older Pentium machines with 1GHz processors; we believe that the performance will be significantly better on newer machines with faster processors.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

However, the advantages of using a new scripting language outweigh the disadvantages. This is because using an existing script might require additional parsers, which may not be suitable for web-based applications. Our scripting language only requires a small parser that can be embedded into client application. Furthermore, the proposed script is sufficient to serve the purpose of defining conditions and assigning animations to objects. It could be extended in future when the need arises.

5.2 RECOMMENDATIONS FOR FURTHER RESEARCH

In this project, we have designed a basic framework that allows authors to create interactive web-based 3D visualizations. This framework could be applied to the publication of online documents, e-Learning documents, and other applications where visualization is desired.

Future research should be focused in the following areas:

- Provide support for importing composite Objects from VRML files and translate these into our data structure. To provide seamless support for VRML, we should allow complex VRML Objects to be loaded into the system. These Objects could be created by a more experienced programmer, or created using VRML authoring tools. Animations and behaviors can then be applied to these Objects. This will create a more realistic and visually appealing scene to improve the visualization effectiveness.

- Provide support for animating several Objects as a group. A more complex Object hierarchy should be designed to allow different Objects to be grouped together. Objects in the same group can then take on the same Animations and behavior collectively.

- Provide support for animation parts of a composite Object as individual Objects. This requires a more complex design than the previous point, but it will be extremely useful. For example, if a composite Object that represents a car is created. The author needs to control the velocity of the composite Object collectively, as well as control the rotation of the ‘wheels’ separately

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

from the 'body' of the car. Ideally, unknown Objects loaded into the scene can also be 'broken down' into simpler Objects automatically to allow authors to assign Animations and behavior.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

PUBLICATIONS

K.S. Neo, Q.P. Lin, Robert K.L. Gay, 'A Web-based System for Interactive Visualization of Scientific Concepts', ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI2004), Singapore, 16-18 June 2004.

Lin Q., Zhang L.,Choo T. F., Wang W., Neo K. S., Gay R., "3D Collaborative Virtual Environment Based eLearning System", accepted for publication in Journal of Applied Systems Studies, Cambridge International Science Publishing, UK.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

BIBLIOGRAPHY

[1] Stokes, S. (2002). "Visual Literacy in Teaching and Learning: A Literature Perspective". Electronic journal for the integration of technology in Education. Idaho State University. Available online: http://ejite.isu.edu/Volume1No1/Stokes.html

[2] Macromedia Flash MX 2004, http://www.macromedia.com/software/flash/.

[3] J. Codling. "Using Macromedia Flash to Produce Rich Multimedia Content on the World Wide Web (v.1.1)". Department of Electronics and Computer Science, University of Southampton. 2002.

[4] Actionscript.org Macromedia Flash Resources and Tutorials. http://www.actionscripts.org/.

[5] 3D Studio Max. http://www.discreet.com/products/3dsmax/.

[6] Ulead COOL 3D. http://www.ulead.com/cool3d/runme.htm.

[7] Blender 3D. http://www.blender3d.com/.

[8] G. Bell, A. Parisi, M. Pesce, "The Virtual Reality Modeling Language. Version 1.0 Specification", 1995. http://www.web3d.org/technicalinfo/specifications/VRML1.0/index.html.

[9] The VRML Consortium, "The Virtual Reality Modeling Language", ISO/IEC DIS 14772-1, 1997. http://www.vrml.org/Specifications/VRML97.

[10] "VRML EAI Pointers". http://tecfa.unige.ch/guides/vrml/links/eai.html.

[11] "VRML 2.0 EAI FAQ". http://www.frontiernet.net/~imaging/eaifaq.html.

[12] Web3D Consortium, X3D Documentation. http://www.web3d.org/x3d/.

[13] Java3D API. http://java.sun.com/products/java-media/3D/.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

[14] The Xj3D Project. http://www.xj3d.org/.

[15] Web3D Consortium - Xj3D Java-based X3D Toolkit & Browser. http://www.web3d.org/x3d/applications/xj3d/.

[16] S.D. Benford, D N. Snowdon, C.C. Brown, G.T. Reynard, and R.J. Ingram, "Visualising and Populating the Web: Collaborative Virtual Environments for Browsing, Searching and Inhabiting Webspace". Proc. 8th Joint European Networking Conference (JENC '97), Edinburgh, UK, 1997.

[17] S.D. Benford, D. Bowers, J. M., Fahlén, L. E., Mariani, J. and Rodden, T. R., "Supporting Co-operative Work in Virtual Environments", The Computer Journal, Oxford University Press, Vol. 37, No. 8, 1994.

[18] J. Locke, "An Introduction to the Internet Networking Environment and SIMNET/DIS", Computer Science Department, Naval Postgraduate School, 1994.

[19] D.B. Anderson, J.W. Barrus, J.H. Howard, C. Rich, C. Shen, R.C. Waters, "Building Multi-User Interactive Multimedia Environments at MERL", IEEE Multimedia, Vol. 2, No. 4, 1995, pp. 77-82.

[20] R. Waters, D. Anderson, J. Barrus, D. Brogan, M. Casey, S. McKeown, T. Nitta, I. Sterns, W. Yerazunis, "Diamond Park and SPLINE: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability", Presence: Teleoperators and Virtual Environments, Vol. 6, No. 4, 1997, pp. 461-480.

[21] C. Greenhalgh, S. Benford, "MASSIVE: a Distributed Virtual Reality System Incorporating Spatial Trading", Proc. 15th International Conference on Distributed Computing Systems, 1995, pp. 27-34.

[22] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, "NPSNET: A Network Software Architecture for Large Scale Virtual Environments", MIT Presence, Vol. 3, No. 4, 1994.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

[23] M.R. Macedonia , D.P. Brutzman , M.J. Zyda , D.R. Pratt , P.T. Barham , J. Falby , J. Locke, "NPSNET: a multi-player 3D virtual environment over the Internet", Proc. 1995 Symp. Interactive 3D graphics, Monterey, California, United States, 1995, pp. 93-94.

[24] M. Capps, D. McGregor, D. Brutzman, M. Zyda, "NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments", IEEE Computer Graphics and Applications Vol. 20, No. 5, 2000, pp. 12-15.

[25] M. Reddy, L. Iverson, Y. Leclerc, "GeoVRML 1.0: Adding Geographic Support to VRML", GeoInformatics, Vol. 3, 2000.

[26] M. Reddy, L. Iverson, Y. Leclerc, A. Heller, "GeoVRML: Open Web-based 3D Cartography", Proc. International Cartographic Conference (ICC2001), Beijing, China, 2001.

[27] GeoVRML.org. http://www.geovrml.org/.

[28] R. Lipman, K. Reed, "Using VRML In Construction Industry Applications", Proc. Web3D: VRML 2000, Monterey, CA, 2000, pp. 1-7.

[29] F. Marir, K. Ouazzane, K. Zerzour, "OSCONVR: An Interactive Virtual Reality Interface to an Object-Oriented Database System for Construction Architectural Design", International Conference on Computational Science (ICCS 2002), 2002, pp. 258-267.

[30] J. Han, J. Kim, D. Lee, J. Lee, H.K. Jong, S.P. Kwang, S.K. Heung, "Three-Dimensional Visualization of Human Vocal Organ on the Web", Journal of Digital Imaging, 2002, pp. 264-266.

[31] B.G. Witmer, J.H. Bailey, B.W. Knerr, "Virtual Spaces and Real World Places: Transfer of Route Knowledge", International Journal of Human–Computer Studies, Vol. 45, No. 4, 1996, pp. 413-428.

[32] B.M. Slator, P. Juell, P.E. McClean, B. Saini-Eidukat, D.P. Schwert, A.R. White, C. Hill, "Virtual Environments for Education", World Conference

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

on Educational Media, Hypermedia and Telecommunications (ED-MEDIA 99), Seattle, WA, 1999, pp. 875-880.

[33] O. Casher, C. Leach, C.S. Page, H.S. Rzepa, "Virtual Reality Modeling Language (VRML) in Chemistry". Chemistry in Britain, Vol. 34, 1998, pp. 26. http://www.ch.ic.ac.uk/rzepa/vrml/

[34] I.L. Ruiz, E.L. Espinosa, G.C. García, M.Á. Gómez-Nieto, "Computer-Assisted Learning of Chemical Experiments through a 3D Virtual Lab", Proc. International Conference on Computational Science-Part I, 2002, pp. 704-712.

[35] C. Bouras, A. Philopoulos, T. Tsiatsos, "E-Learning through distributed virtual environments", Journal of Network and Computer Applications, Academic Press, No. 3, Vol. 24, 2001, pp. 175-199.

[36] J. Mathews, D. Leisawitz, J. Thieman, "The Interactive Universe: An Application of VRML and the Web to Science", NSSDC News, Vol. 13, No. 4, 1997.

[37] M.D. Lytras, A. Pouloudi, "E-learning: Just a waste of time", In D. Strong, D. Straub and J.I. DeGross, editors, 7th Americas Conference on Information Systems (AMCIS 2001), 2001, pp. 216-222.

[38] M. Pinheiro, J. Lima, N. Edelweiss, N. Layaïda, T. Lemlouma, "An Open E-Learning Authoring Environment", Spezielles Seminar im WS 2002/2003 - E-Learning.

[39] T.G. Kirner, C. Kirner, A.L.S. Kawamoto, J. Cantão, A. Pinto, R.S. Wazlawick, "Development of a Collaborative Virtual Environment for Educational Applications", Proc. Sixth International Conference on 3D Web Technology, Paderbon, Germany, 2001, pp. 61-68.

[40] G. Proctor, C. Winter. "Information Flocking: Data Visualization in Virtual Worlds Using Emergent Behaviors", In J.-C. Heudin, editor, Proc. 1st International Conference on Virtual Worlds, Springer-Verlag, Berlin, 1998, pp. 168-176.

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

[41] C. Cavallar, D. Dögl, "Organizing Information in 3D". J.-C. Heudin(Ed.): Virtual Worlds 98, 1998, pp. 308-314.

[42] M. Kreuseler, N. Lopez, H. Schumann, "A Scalable Framework for Information Visualization", Proc. IEEE Symp. Information Vizualization 2000, 2000, pp. 27.

[43] F.S. Tamiosso, A.B. Raposo, L.P. Magalhaes, I.L.M. Ricarte, "Building Interactive Animations using VRML and Java", Brazilian Symp. Computer Graphics and Image Processing (SIBGRAPHI '97), 1997, pp. 42-48.

[44] N. Megnenat-Thalmann, D. Thalmann, "Computer Animation", Springer, Wien, 1985.

[45] W.M. Lee, M.G Lee, "An Animation Toolkit Based on Motion Mapping", Computer Graphics International (CGI'00), 2000, pp. 11-18.

[46] F.S. Tamiosso, A.B.Raposo, L.P. Magalhaes, I.L.M. Ricarte, "Building Interactive Animations Using VRML and Java", Proc. X Brazilian Symposium on Computer Graphics and Image Processing, 1997, pp. 42-48.

[47] S. Diehl, "Object-Oriented Animations with VRML++", Proc. Virtual Environment 98 Conference and 4th Eurographics Workshop, Stuttgart, Germany, 1998.

[48] D. Vodislav, M. Vazirgiannis, "Structured Interactive Animation for Multimedia Documents", IEEE International Symp. Visual Languages (VL'00), 2000, pp. 95-96.

[49] M. Duecker, G. Lehrenfeld, W. Mueller, C. Tahedl, "A Generic System for Interactive Real-Time Animation", 1997 Workshop on Engineering of Computer-Based Systems (ECBS '97), 1997, pp. 263-270.

[50] CONTIGRA Project web pages: http://www.contigra.com

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

[51] R. Dachselt, M. Hinz, K. Meiner, "CONTIGRA: An XML-Based Architecture for Component-Oriented 3D Applications", Proc. ACM Web3D 2002 Symposium, Tempe (USA), 2002, pp. 24-28.

[52] R. Dachselt, E. Rukzio, "Behavior3D: an XML-Based Framework for 3D Graphics Behavior", Proc. Eighth International Conference on 3D Web Technology, Saint Malo, France, 2003, pp. 101 – 112.

[53] Alice – Learn to Program Interactive 3D Graphics. http://www.alice.org/

[54] M. Conway, S. Audia, T. Burnette, D. Cosgrove, K. Christiansen, R. Deline, J. Durbin, R. Gossweiler, S. Koga, C. Long, B. Mallory, S. Miale, K. Monkaitis, J. Patten, J. Pierce, J. Shochet, D. Staack, B. Stearns, R. Stoakley, C. Sturgill, J. Viega, J. White, G. Williams, R. Pausch, "Alice: Lessons Learned from Building a 3D System for Novices", Proc. SIGCHI conference on Human Factors in Computing Systems, 2000, pp. 486-493.

[55] R. Pausch (head), T. Burnette, A.C. Capehart, M. Conway, D. Cosgrove, R. DeLine, J. Durbin, R. Gossweiler, S. Koga, J. White, "Alice: Rapid Prototyping for Virtual Reality". IEEE Computer Graphics and Applications, 1995, pp. 8-11.

[56] W3C. "Synchronized Multimedia Integration Language (SMIL 2.0)". http://www.w3.org/TR/smil20/

[57] Marc H. Brown, "Perspectives on Algorithm Animation", Proc. ACM SIGCHI'88 Conference on Human Factors in Computing Systems, Washington D.C., 1998, pp. 33-38.

[58] Stasko, J.T., "Using Student-Built Algorithm Animations As Learning Aids", 28th SIGCSE Technical Symposium on Computer Science Education, San Jose, California, 1997, pp. 25-29.

[59] Jsamba – Java Version of the SAMBA Animation Program. http://www.cc.gatech.edu/gvu/softviz/algoanim/jsamba/

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

[60] Pierson W.C., Rodger S.H., "Web-based animation of data structures using JAWAA", 29th SIGCSE Technical Symposium on Computer Science Education, 1998, pp. 267-271.

[61] A. Akingbade, T. Finley, D. Jackson, P. Patel, and S.H. Rodger, "JAWAA: Easy Web-Based Animation from CS 0 to Advanced CS Courses", Proc. 34 th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada, 2003, pp. 162-166.

[62] G. Roling and B. Freisleben, "AnimalScript: An Extensible Scripting Language for Algorithm Animation", 32nd ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2001), Charlotte, North Carolina, February 2001, pp. 70-74.

[63] Easy GIF Animator, http://www.bluementals.net/egifan/

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

Appendix A Client Application Source Code

In this section, we present the source code for the six significant components – TreeControl.java, ObjInfo.java, scriptParser.java, VRMLBuilder.java, VRMLObject.java, and clientConnect.java.

```
//---------------TreeControl.java---------------//

import java.awt.*;
import java.io.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;
import java.util.Hashtable;
import java.util.Enumeration;
import java.net.*;

// W3C DOM classes
import org.w3c.dom.*;

//xerces parser
import org.apache.xerces.parsers.DOMParser;

// JAXP's classes for DOM I/O
import javax.xml.parsers.*;

//permissions for file io
import java.security.*;
import com.ms.security.*;
import com.ms.security.permissions.*;

//VRML
import vrml.external.Browser;

public class TreeControl extends JPanel implements ScrollPaneConstants
{
    protected DefaultMutableTreeNode rootNode, rootAni, rootObj, rootDesc;
    protected DefaultTreeModel treeModel;
    protected JTree tree;
    private Toolkit toolkit = Toolkit.getDefaultToolkit();

    /* These three members are a part of the JAXP API and are used to parse the XML
    * text into a DOM object (of type Document).
    */
        private DocumentBuilderFactory dbf;
    private DocumentBuilder db;
    private Document doc;

    //xerces DOM parser
    private DOMParser parser;

     mainEditor parentApplet;
        //hashtable
    Hashtable ObjectsTable = new Hashtable(); //stores list of objects in the tree
    Hashtable ObjectTable = new Hashtable(); //stores object name and pointer to object
    Hashtable AniTable = new Hashtable(); //stores list of animations in the tree
    Hashtable AniChoiceTable = new Hashtable(); //stores list of animations with name as key

    //store description
    DescInfo descInfo = new DescInfo();

    //build VRML
    VRMLBuilder vrmlBuild;

    //for reading xml file
    String cache;
    String dir;
    String ipAddress;
    String port;
    String fn;

    //constructor
    public TreeControl ()
    {

    }

    //initialize
    public void initialize(boolean loadXML,String fn, mainEditor parent)
                throws ParserConfigurationException
    {
        BufferedReader reader;
        BufferedWriter writer;
        String line;
        StringBuffer xmlText;
        this.vrmlBuild = parent.vrmlBuild;

        //hashtables
        this.AniChoiceTable = parent.AniChoiceTable;
        this.AniTable = parent.AniTable;
        this.ObjectsTable = parent.ObjectsTable;
        this.ObjectTable = parent.ObjectTable;
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
this.parentApplet = parent;
this.cache = parentApplet.cacheDir;
this.dir = parentApplet.dir;
this.ipAddress = parentApplet.ipAddress;
this.port = parentApplet.port;
this.fn = parentApplet.fn;

//create a brand new tree
rootNode = new DefaultMutableTreeNode("Root Node");
treeModel = new DefaultTreeModel(rootNode);

rootAni = addObject(rootNode, "Animation");
rootObj = addObject(rootNode, "Object");
rootDesc = addObject(rootNode, descInfo);

treeModel.addTreeModelListener(new MyTreeModelListener());

tree = new JTree(treeModel);
tree.setEditable(false);
tree.getSelectionModel().setSelectionMode (TreeSelectionModel.SINGLE_TREE_SELECTION);
tree.setShowsRootHandles(true);

JScrollPane scrollPane = new JScrollPane
            (tree,VERTICAL_SCROLLBAR_ALWAYS,HORIZONTAL_SCROLLBAR_ALWAYS);
setLayout(new GridLayout(1,0));
add(scrollPane);

if (loadXML == true)
{
    try
    {
        com.ms.security.PolicyEngine.assertPermission(com.ms.security.PermissionID.PROPERTY);
        com.ms.security.PolicyEngine.assertPermission(com.ms.security.PermissionID.UI);
        com.ms.security.PolicyEngine.assertPermission(com.ms.security.PermissionID.FILEIO);
        //try reading from server
        URL url;
        URLConnection urlConn;

        //for reading and writing on local system
        File f = new File(System.getProperty("user.home")
                + System.getProperty("file.separator")
                + fn);
        System.out.println(f.getAbsolutePath() + " created");
        FileWriter fw = new FileWriter(f);
        FileReader fr = new FileReader(f);

        url = new URL(parentApplet.getCodeBase().toString() + dir + "/" + fn);
        urlConn = url.openConnection();
        urlConn.setDoInput(true);
        urlConn.setUseCaches(false);

        reader = new BufferedReader(new InputStreamReader(urlConn.getInputStream()));
        //write into temporary file
        writer = new BufferedWriter(fw);
        xmlText = new StringBuffer();

        while ((line = reader.readLine()) != null)
        {
            xmlText.append(line.trim());
            writer.write(line);
            writer.newLine();
        }

        reader.close();
        writer.close();

        //read from the temp file
        if ( fn.substring( fn.indexOf( '.' ) ).equals( ".xml" ) )
        {

            reader = new BufferedReader(fr);
            xmlText = new StringBuffer();

            while ( ( line = reader.readLine() ) != null )
            {
                xmlText.append(line.trim());
            }

            // The file will have to be re-read when the Document object is parsed
            reader.close();

            // Begin by initializing the object's DOM parsing objects
            dbf = DocumentBuilderFactory.newInstance();
            dbf.setValidating( false );
            db = dbf.newDocumentBuilder();

            Node grandfather = parseXml(f.getAbsolutePath());
            NodeList childrenNodes = grandfather.getChildNodes();
            Node father;

            if (childrenNodes != null)
            {
                for (int i = 0; i < childrenNodes.getLength(); i ++)
                {
                    father = childrenNodes.item(i);

                    if (father.getNodeName().equals("Animation"))
                    {
                        getDataFromDOM(father, true);
                    } else if ( father.getNodeName().equals("Object"))
                    {
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                            getDataFromDOM(father, false);
                        } else if (father.getNodeName().equals("Description"))
                        {
                            String desc = getGrandChild(father);
                            descInfo.setDesc(desc);
                            System.out.println("Description = " + desc);
                        }
                    }
                }
            }
        } catch  (MalformedURLException mue)
        {
            System.out.println("MalformedURLException " + mue.toString());
        } catch (IOException ioe)
        {
            System.out.println("IOException " + ioe.toString());
        } catch( Exception ex ) {
            ex.printStackTrace();
        }
    }
}

//add new animation subtree
public void addNewAni(AniInfo newAni)
{
    DefaultMutableTreeNode Animation;

    Animation = addObject(rootAni,newAni);

    //process animation functions
    newAni.processFunctions();
}

//add new object subtree
public void addNewObj(ObjInfo newObj)
{
    DefaultMutableTreeNode newObject;
    newObject = addObject(rootObj,newObj);

    //process scripts
    if (newObj.script.trim().equals("") == false)
    {
        scriptParser parse = new scriptParser(newObj);
        String[] strParsed = parse.getTokens(newObj.script);
        boolean result = parse.processTokens(strParsed);
        if (result == true)
        {
            //save the extracted data now
            newObj.IFexp = parse.IFexp;
            newObj.IFstmt = parse.IFstmt;
            newObj.ELSEIFexp = parse.ELSEIFexp;
            newObj.ELSEIFstmt = parse.ELSEIFstmt;
            newObj.ELSEstmt = parse.ELSEstmt;

        } else
        {
            System.out.println("************ERROR************");
            System.out.println("** please check the syntax **");
            System.out.println("******************************");
        }
    }
    //create new vrml object
    VRMLObject vrmlNew = vrmlBuild.VRMLGen(newObj);
}

/** Remove all nodes except the root node. */
public void clear()
{
    rootAni.removeAllChildren();
    rootObj.removeAllChildren();
    treeModel.reload();
    vrmlBuild.removeAll(parentApplet.ObjectsTable);
    parentApplet.AniChoiceTable.clear();
    parentApplet.AniTable.clear();
    parentApplet.ObjectsTable.clear();
    //remove scene description
    this.descInfo.setDesc("");
}

/** save the tree onto XML file */
public void save()
{
    //creates the DOM structure from the jtree and stores it in a temp XML
    //file before submitting to server
    DocumentBuilderFactory dbf1;
    DocumentBuilder db1;
    Document doc1;

    try
    {
        dbf1 = DocumentBuilderFactory.newInstance ();
        db1 = dbf1.newDocumentBuilder ();
        doc1 = db1.newDocument ();
        Node root = createXMLNode (doc1);
        doc1.appendChild(root);

        //write the document to a temporary file
        XMLOutput xmlOut = new XMLOutput();
        boolean isSaved = xmlOut.output(doc1, ipAddress, port, fn);
        if (isSaved)
        {
            JOptionPane.showMessageDialog(parentApplet, "File saved successfully :)");
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
            } else
            {
                JOptionPane.showMessageDialog(parentApplet, "Error occurred while saving file");
            }
        } catch (ParserConfigurationException pe)
        {
            System.out.println("ParserConfigurationException :'D " + pe.toString());
        }
    }

    /** Edit leaf node **/
    public void editNode()
    {
        //check active node
        //if it's a leaf node, pop up window for editing parameter
        ObjInfo curObject;
        AniInfo curAni;

        TreePath currentSelection = tree.getSelectionPath();
        if (currentSelection != null)
        {
            DefaultMutableTreeNode currentNode = (DefaultMutableTreeNode)
                                                 (currentSelection.getLastPathComponent());
            MutableTreeNode parent = (MutableTreeNode) (currentNode.getParent());
            if (parent != null)
            {
                if (currentNode.getParent() == rootNode)
                {
                    if (currentNode == rootDesc)
                    {
                        frameDescEditor descEdit = new frameDescEditor(descInfo);
                        descEdit.show();
                    } else
                    {
                        //ask user to select an object/animation
                        String Msg = "please choose an animation or object to edit";

                        JOptionPane.showMessageDialog(parentApplet, Msg);
                    }
                } else
                {
                    //get the name of the parent of this subtree
                    //(eg: Animation3, Object5 etc)
                    //and check if object or animation is selected
                    String subtreeParent = currentSelection.getPathComponent(1).toString();

                    if (subtreeParent.startsWith("Object"))
                    {
                        //object subtree
                        //get the data from each leaf node
                        DefaultMutableTreeNode objNode = (DefaultMutableTreeNode)
                                                         currentSelection.getLastPathComponent();
                        curObject = (ObjInfo)objNode.getUserObject();
                        int pos = curObject.pos;
                        frameObjectEditor objEdit = new frameObjectEditor(curObject);
                        objEdit.show();
                    } else if (subtreeParent.startsWith("Anim")){
                        //animation subtree
                        //object subtree
                        //get the data from each leaf node
                        DefaultMutableTreeNode objNode = (DefaultMutableTreeNode)
                                                         currentSelection.getLastPathComponent();
                        curAni = (AniInfo)objNode.getUserObject();
                        int pos = curAni.pos;
                        frameAniEditor aniEdit = new frameAniEditor(curAni);
                        aniEdit.show();
                    }
                }
            }
            return;
        }
    }

    /** Remove the currently selected node. */
    public void removeCurrentNode()
    {
        //check active node
        //if it's a leaf node, pop up window for editing parameter
        DefaultMutableTreeNode objNode;

        TreePath currentSelection = tree.getSelectionPath();
        if (currentSelection != null)
        {
            DefaultMutableTreeNode currentNode = (DefaultMutableTreeNode)
                                                 (currentSelection.getLastPathComponent());
            MutableTreeNode parent = (MutableTreeNode) (currentNode.getParent());
            if (parent != null)
            {
                if (currentNode.getParent() == rootNode)
                {
                    //ask user to select an object/animation
                    JOptionPane.showMessageDialog(parentApplet, "please choose an animation or object to
remove");
                } else
                {
                    //get the name of the parent of this subtree
                    //(eg: Animation3, Object5 etc)
                    //and check if object or animation is selected
                    String subtreeParent = currentSelection.getPathComponent(1).toString();

                    if (subtreeParent.startsWith("Object"))
                    {
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                        //object subtree
                        //get the data from each leaf node
                        objNode = (DefaultMutableTreeNode) currentSelection.getLastPathComponent();
                        removeObj(objNode);

                    } else if (subtreeParent.startsWith("Anim"))
                    {
                        objNode = (DefaultMutableTreeNode) currentSelection.getLastPathComponent();
                        removeAni(objNode);
                    }
                }
            }
            return;
        }

        // Either there was no selection, or the root was selected.
        toolkit.beep();
    }

    //for adding general node
    public DefaultMutableTreeNode addObject(DefaultMutableTreeNode parent,
                                            String child)
    {

        DefaultMutableTreeNode childNode = new DefaultMutableTreeNode(child);

        if (parent == null)
        {
            parent = rootNode;
        }

        treeModel.insertNodeInto(childNode, parent, parent.getChildCount());

        return childNode;
    }

    //for adding object subtree to jtree
    public DefaultMutableTreeNode addObject(DefaultMutableTreeNode parent, ObjInfo child)
    {
        DefaultMutableTreeNode childNode = new DefaultMutableTreeNode(child);

        if (parent == null)
        {
            parent = rootNode;
        }

        treeModel.insertNodeInto(childNode, parent,
                                 parent.getChildCount());
        return childNode;
    }

    //for adding animation subtree to jtree
    public DefaultMutableTreeNode addObject(DefaultMutableTreeNode parent,
                                            AniInfo child)
    {

        DefaultMutableTreeNode childNode = new DefaultMutableTreeNode(child);

        if (parent == null)
        {
            parent = rootNode;
        }

        treeModel.insertNodeInto(childNode, parent, parent.getChildCount());
        return childNode;
    }

    //for adding description to jtree
    public DefaultMutableTreeNode addObject(DefaultMutableTreeNode parent,
                                            DescInfo child)
    {

        DefaultMutableTreeNode childNode = new DefaultMutableTreeNode(child);

        if (parent == null)
        {
            parent = rootNode;
        }

        treeModel.insertNodeInto(childNode, parent, parent.getChildCount());
        return childNode;
    }

    private void removeAni(DefaultMutableTreeNode removeNode)
    {
        //update aniTable and aniChoiceTable
        this.AniChoiceTable = parentApplet.AniChoiceTable;
        this.AniTable = parentApplet.AniTable;

        //find the position and name of this ani to be removed
        AniInfo aniInfo = (AniInfo)removeNode.getUserObject();
        int pos = aniInfo.pos;
        String aniName = aniInfo.aniName;

        //remove the ani from AniChoiceTable and AniTable
        //recoganize ani in the tables
        if (  (AniTable.size()!=1) && (pos<AniTable.size()) )//do while not the last object unless is 1st
row
        {
            for(int i=pos+1; i<=AniTable.size(); i++)
            {
                AniInfo AniObj =(AniInfo)AniTable.get(new Integer(i));
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
            AniObj.pos = i-1;
            AniObj.setString();
            AniTable.put(new Integer(i-1), AniObj);
        }
        //remove last object, since all Treeobject move up
        AniTable.remove(new Integer(AniTable.size()));
    }
    else //size=1 or last row
    {
        AniTable.remove(new Integer(pos));
    }

    //delete information from AniChoiceTable
    AniChoiceTable.remove(new String(aniName));

    //remove tree node
    treeModel.removeNodeFromParent(removeNode);

    //refresh jtree display

}

private void removeObj(DefaultMutableTreeNode removeNode)
{
    //update ObjectsTable and VRMLObjectsTable
    this.ObjectsTable = parentApplet.ObjectsTable;
    this.ObjectTable = parentApplet.ObjectTable;

    //get the ID of the object selected
    ObjInfo objectInfo = (ObjInfo)removeNode.getUserObject();
    int pos = objectInfo.pos;

    vrmlBuild.removeObject(objectInfo);

    ObjectTable.remove(objectInfo.objName);

    //remove the obj from ObjectsTable
    //reorganize obj in the table
    if (  (ObjectsTable.size()!=1) && (pos<ObjectsTable.size()) )
    //do while not the last object unless is 1st row
    {
        for(int i=pos+1; i<=ObjectsTable.size(); i++)
        {
            ObjInfo Obj =(ObjInfo)ObjectsTable.get(new Integer(i));
            Obj.pos = i-1;
            Obj.setString();
            ObjectsTable.put(new Integer(i-1), Obj);
        }

        //remove last object, since all Treeobject move up
        ObjectsTable.remove(new Integer(ObjectsTable.size()));

    } else//size=1 or last row
    {
        ObjectsTable.remove(new Integer(pos));
    }

    //remove tree node
    treeModel.removeNodeFromParent(removeNode);

}

class MyTreeModelListener implements TreeModelListener
{
    public void treeNodesChanged(TreeModelEvent e)
    {
        DefaultMutableTreeNode node;
        node = (DefaultMutableTreeNode) (e.getTreePath().getLastPathComponent());

        /*
         * If the event lists children, then the changed
         * node is the child of the node we've already
         * gotten.  Otherwise, the changed node and the
         * specified node are the same.
         */
        try
        {
            int index = e.getChildIndices()[0];
            node = (DefaultMutableTreeNode) (node.getChildAt(index));
        } catch (NullPointerException exc) {}

    }
    public void treeNodesInserted(TreeModelEvent e)
    {
    }
    public void treeNodesRemoved(TreeModelEvent e)
    {
    }
    public void treeStructureChanged(TreeModelEvent e)
    {
    }
}

private void getDataFromDOM(Node root, boolean isAni)
{
    AniInfo newAni;
    ObjInfo newObj;
    int pos;

    String name;
    // Get data from root node
    name = root.getNodeName();
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
//recurse child nodes
if(root.hasChildNodes())
{
    NodeList children;
    int numChildren;
    Node node;
    String nodeName;
    String data;

    children = root.getChildNodes();

    // Only recurse if Child Nodes are non-null
    if( children != null )
    {
        numChildren = children.getLength();

        if (isAni)
        {
            newAni = new AniInfo();
            pos = parentApplet.AniTable.size() + 1;
            newAni.pos = pos;
            newAni.setString();
            for (int i = 0; i < numChildren; i ++)
            {
                node = children.item(i);
                if( node != null )
                {
                    // A special case could be made for each Node type.
                    if( node.getNodeType() == Node.ELEMENT_NODE )
                    {
                        nodeName = node.getNodeName();
                        if (nodeName.equals("aniName"))
                        {
                            data = getGrandChild(node);
                            newAni.aniName = data;
                        } else if (nodeName.equals("Pos_x"))
                        {
                            data = getGrandChild(node);
                            newAni.Pos_x = data;
                        } else if (nodeName.equals("Pos_y"))
                        {
                            data = getGrandChild(node);
                            newAni.Pos_y = data;
                        } else if (nodeName.equals("Pos_z"))
                        {
                            data = getGrandChild(node);
                            newAni.Pos_z = data;
                        } else if (nodeName.equals("Rot_x"))
                        {
                            data = getGrandChild(node);
                            newAni.Rot_x = data;
                        } else if (nodeName.equals("Rot_y"))
                        {
                            data = getGrandChild(node);
                            newAni.Rot_y = data;
                        } else if (nodeName.equals("Rot_z"))
                        {
                            data = getGrandChild(node);
                            newAni.Rot_z = data;
                        } else if (nodeName.equals("Rot_angle"))
                        {
                            data = getGrandChild(node);
                            newAni.Rot_angle = data;
                        } else if (nodeName.equals("Scale_x"))
                        {
                            data = getGrandChild(node);
                            newAni.Scale_x = data;
                        } else if (nodeName.equals("Scale_y"))
                        {
                            data = getGrandChild(node);
                            newAni.Scale_y = data;
                        } else if (nodeName.equals("Scale_z"))
                        {
                            data = getGrandChild(node);
                            newAni.Scale_z = data;
                        } else if (nodeName.equals("Color_R"))
                        {
                            data = getGrandChild(node);
                            newAni.Color_R = data;
                        } else if (nodeName.equals("Color_G"))
                        {
                            data = getGrandChild(node);
                            newAni.Color_G = data;
                        } else if (nodeName.equals("Color_B"))
                        {
                            data = getGrandChild(node);
                            newAni.Color_B = data;
                        }
                    }
                }
                //after collecting all relevant information, add new reference
                //to animation hashtables
                parentApplet.AniChoiceTable.put(newAni.aniName,newAni);
                parentApplet.AniTable.put(new Integer(pos),newAni);
                addNewAni(newAni);
            } else if (!isAni)
            {
                newObj = new ObjInfo(this.parentApplet);
                pos = parentApplet.ObjectTable.size() + 1;
                newObj.pos = pos;
                newObj.setString();
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
for (int i = 0; i < numChildren; i ++)
{
    node = children.item(i);
    if( node != null )
    {
        // A special case could be made for each Node type.
        if( node.getNodeType() == Node.ELEMENT_NODE )
        {
            nodeName = node.getNodeName();
            if (nodeName.equals("objName"))
            {
                data = getGrandChild(node);
                newObj.objName = data;
            } else if (nodeName.equals("objType"))
            {
                data = getGrandChild(node);
                newObj.objType = Integer.parseInt(data);
            } else if (nodeName.equals("D_radius"))
            {
                data = getGrandChild(node);
                newObj.D_radius = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("D_height"))
            {
                data = getGrandChild(node);
                newObj.D_height = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("D_x"))
            {
                data = getGrandChild(node);
                newObj.D_x = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("D_y"))
            {
                data = getGrandChild(node);
                newObj.D_y = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("D_z"))
            {
                data = getGrandChild(node);
                newObj.D_z = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("D_top"))
            {
                data = getGrandChild(node);
                newObj.D_top = checkBoolean(data);
            } else if (nodeName.equals("D_side"))
            {
                data = getGrandChild(node);
                newObj.D_side = checkBoolean(data);
            } else if (nodeName.equals("D_bottom"))
            {
                data = getGrandChild(node);
                newObj.D_bottom = checkBoolean(data);
            } else if (nodeName.equals("pos_x"))
            {
                data = getGrandChild(node);
                newObj.pos_x = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("pos_y"))
            {
                data = getGrandChild(node);
                newObj.pos_y = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("pos_z"))
            {
                data = getGrandChild(node);
                newObj.pos_z = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("rot_x"))
            {
                data = getGrandChild(node);
                newObj.rot_x = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("rot_y"))
            {
                data = getGrandChild(node);
                newObj.rot_y = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("rot_z"))
            {
                data = getGrandChild(node);
                newObj.rot_z = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("rot_angle"))
            {
                data = getGrandChild(node);
                newObj.rot_angle = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("scale_x"))
            {
                data = getGrandChild(node);
                newObj.scale_x = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("scale_y"))
            {
                data = getGrandChild(node);
                newObj.scale_y = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("scale_z"))
            {
                data = getGrandChild(node);
                newObj.scale_z = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("color_R"))
            {
                data = getGrandChild(node);
                newObj.color_R = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("color_G"))
            {
                data = getGrandChild(node);
                newObj.color_G = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("color_B"))
            {
                data = getGrandChild(node);
                newObj.color_B = Float.valueOf(data).floatValue();
            } else if (nodeName.equals("script"))
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                                {
                                    data = getGrandChild(node);
                                    newObj.script = data;
                                } else if (nodeName.equals("text"))
                                {
                                    data = getGrandChild(node);
                                    newObj.text = data;
                                }

                            }
                        }
                    }
                    //after collecting all relevant information, add new reference
                    //to object hashtables
                    parentApplet.ObjectTable.put(newObj.objName,newObj);
                    parentApplet.ObjectsTable.put(new Integer(pos),newObj);

                    addNewObj(newObj);
                }
            }
        }
    }

    //check boolean value of string
    private boolean checkBoolean(String data)
    {
        if (data.trim().toLowerCase().equals("true"))
        {
            return true;
        } else
        {
            return false;
        }
    }

    //check boolean value of string
    private String checkBoolean(boolean data)
    {
        if (data == true)
        {
            return "true";
        } else
        {
            return "false";
        }
    }

    public static String getGrandChild(Node node)
    {
        StringBuffer buf = new StringBuffer();
        NodeList children = node.getChildNodes();
        for (int i = 0; i < children.getLength(); i++)
        {
            Node textChild = children.item(i);
            if (textChild.getNodeType() != Node.TEXT_NODE)
            {
                System.err.println("Mixed content! Skipping child element " + textChild.getNodeName());
                continue;
            }
            buf.append(textChild.getNodeValue());
        }
        return buf.toString();
    }

   /**
    * This method returns a string representing the type of node passed in.
    *
    * @param node org.w3c.Node.Node
    *
    * @return Returns a String representing the node type
    */
   private String getNodeType( Node node )
   {
        String type;
        switch( node.getNodeType() )
        {
            case Node.ELEMENT_NODE:
            {
                type = "Element";
                break;
            }
            case Node.ATTRIBUTE_NODE:
            {
                type = "Attribute";
                break;
            }
            case Node.TEXT_NODE:
            {
                type = "Text";
                break;
            }
            case Node.CDATA_SECTION_NODE:
            {
                type = "CData section";
                break;
            }
            case Node.ENTITY_REFERENCE_NODE:
            {
                type = "Entity reference";
                break;
            }
            case Node.ENTITY_NODE:
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
            {
                type = "Entity";
                break;
            }
            case Node.PROCESSING_INSTRUCTION_NODE:
            {
                type = "Processing instruction";
                break;
            }
            case Node.COMMENT_NODE:
            {
                type = "Comment";
                break;
            }
            case Node.DOCUMENT_NODE:
            {
                type = "Document";
                break;
            }
            case Node.DOCUMENT_TYPE_NODE:
            {
                type = "Document type";
                break;
            }
            case Node.DOCUMENT_FRAGMENT_NODE:
            {
                type = "Document fragment";
                break;
            }
            case Node.NOTATION_NODE:
            {
                type = "Notation";
                break;
            }
            default:
            {
                type = "???";
                break;
            }
        }
        return type;
    }

    /**
     * This method performs the actual parsing of the XML text
     *
     * @param text A String representing an XML document
     * @return Returns an org.w3c.Node.Node object
     */
    private Node parseXml( String text )
    {
        PolicyEngine.assertPermission(PermissionID.FILEIO);

        try
        {
            parser = new DOMParser();
            parser.parse(text.trim());
            doc = parser.getDocument();
        }
        catch ( Exception e )
        {
            e.printStackTrace();
            System.exit(0);
        }
        return (Node)doc.getDocumentElement();
    }

    // Create Document from tree structure
    /**
    * This takes a tree Node and recurses through the children until each one is added
    * to a DOM element.
    *
    * @return Returns a DefaultMutableTreeNode object based on the root Node passed in
    */
    public Node createXMLNode(Document dc)
    {
        int i = 0;
        AniInfo curAni = null;
        ObjInfo curObj = null;
        Element GrandfatherNode = null; //<root>
        Element FatherNode = null; // <Animation> or <Object>
        Element ChildNode = null; // eg <objName> etc...
        Text txt = null; // data

        GrandfatherNode = dc.createElement("root");

        while ( i < AniTable.size())
        {
            curAni = (AniInfo) AniTable.get(new Integer(i+1));

            //create one element node for each animation
            FatherNode = dc.createElement("Animation");
            GrandfatherNode.appendChild(FatherNode);
            //next create nodes for storing other information
            ChildNode = dc.createElement("aniName");
            txt = dc.createTextNode(curAni.aniName);
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);

            //Position
            ChildNode = dc.createElement("Pos_x");
            txt = dc.createTextNode(curAni.Pos_x);
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Pos_y");
    txt = dc.createTextNode(curAni.Pos_y);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Pos_z");
    txt = dc.createTextNode(curAni.Pos_z);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);

    //Rotation
    ChildNode = dc.createElement("Rot_x");
    txt = dc.createTextNode(curAni.Rot_x);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Rot_y");
    txt = dc.createTextNode(curAni.Rot_y);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Rot_z");
    txt = dc.createTextNode(curAni.Rot_z);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Rot_angle");
    txt = dc.createTextNode(curAni.Rot_angle);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);

    //Scale
    ChildNode = dc.createElement("Scale_x");
    txt = dc.createTextNode(curAni.Scale_x);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Scale_y");
    txt = dc.createTextNode(curAni.Scale_y);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Scale_z");
    txt = dc.createTextNode(curAni.Scale_z);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);

    //Color
    ChildNode = dc.createElement("Color_R");
    txt = dc.createTextNode(curAni.Color_R);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Color_G");
    txt = dc.createTextNode(curAni.Color_G);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("Color_B");
    txt = dc.createTextNode(curAni.Color_B);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);

    i++;
}

//reset i = 0
i = 0;
while ( i < ObjectsTable.size())
{
    curObj = (ObjInfo) ObjectsTable.get(new Integer(i+1));
    //create one element node for each animation
    FatherNode = dc.createElement("Object");
    GrandfatherNode.appendChild(FatherNode);
    //next create nodes for storing other information
    ChildNode = dc.createElement("objName");
    txt = dc.createTextNode(curObj.objName);
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("objType");
    txt = dc.createTextNode(Integer.toString(curObj.objType));
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);

    //Dimension
    ChildNode = dc.createElement("D_radius");
    txt = dc.createTextNode(Float.toString(curObj.D_radius));
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("D_height");
    txt = dc.createTextNode(Float.toString(curObj.D_height));
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("D_x");
    txt = dc.createTextNode(Float.toString(curObj.D_x));
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("D_y");
    txt = dc.createTextNode(Float.toString(curObj.D_y));
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("D_z");
    txt = dc.createTextNode(Float.toString(curObj.D_z));
    ChildNode.appendChild(txt);
    FatherNode.appendChild(ChildNode);
    ChildNode = dc.createElement("D_top");
    txt = dc.createTextNode(checkBoolean(curObj.D_top));
    ChildNode.appendChild(txt);
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("D_side");
            txt = dc.createTextNode(checkBoolean(curObj.D_side));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("D_bottom");
            txt = dc.createTextNode(checkBoolean(curObj.D_bottom));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);

            //Position
            ChildNode = dc.createElement("pos_x");
            txt = dc.createTextNode(Float.toString(curObj.pos_x));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("pos_y");
            txt = dc.createTextNode(Float.toString(curObj.pos_y));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("pos_z");
            txt = dc.createTextNode(Float.toString(curObj.pos_z));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);

            //Rotation
            ChildNode = dc.createElement("rot_x");
            txt = dc.createTextNode(Float.toString(curObj.rot_x));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("rot_y");
            txt = dc.createTextNode(Float.toString(curObj.rot_y));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("rot_z");
            txt = dc.createTextNode(Float.toString(curObj.rot_z));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("rot_angle");
            txt = dc.createTextNode(Float.toString(curObj.rot_angle));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);

            //Scale
            ChildNode = dc.createElement("scale_x");
            txt = dc.createTextNode(Float.toString(curObj.scale_x));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("scale_y");
            txt = dc.createTextNode(Float.toString(curObj.scale_y));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("scale_z");
            txt = dc.createTextNode(Float.toString(curObj.scale_z));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);

            //Color
            ChildNode = dc.createElement("color_R");
            txt = dc.createTextNode(Float.toString(curObj.color_R));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("color_G");
            txt = dc.createTextNode(Float.toString(curObj.color_G));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);
            ChildNode = dc.createElement("color_B");
            txt = dc.createTextNode(Float.toString(curObj.color_B));
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);

            //Script
            ChildNode = dc.createElement("script");
            txt = dc.createTextNode(curObj.script);
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);

            //Text
            ChildNode = dc.createElement("text");
            txt = dc.createTextNode(curObj.text);
            ChildNode.appendChild(txt);
            FatherNode.appendChild(ChildNode);

            i++;
        }

        //create a node for description

        FatherNode = dc.createElement("Description");
        GrandfatherNode.appendChild(FatherNode);
        txt = dc.createTextNode(descInfo.getDesc());
        FatherNode.appendChild(txt);

        return GrandfatherNode;

    }
}
```

//--------------ObjInfo.java--------------//

```
import vrml.external.Node;
import vrml.external.field.*;
import java.util.StringTokenizer;
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
//store user-defined object information
public class ObjInfo
{
    private mainEditor parent;
    private VRMLBuilder vrmlBuild;

    String name = "";
    public VRMLObject vrmlObj = null; //links to VRMLObject corresponding to this object
    public int pos; //object ID
    public String objName = ""; //name
    public int objType = 0; //type: cylinder, cone, sphere, box, text

    //script
    public String script = " "; // the raw string
    public String [][] IFexp;             //  \
    public String [][] IFstmt;            //   |
    public String [][][] ELSEIFexp;       //   | ==> stores extracted information
    public String [][][] ELSEIFstmt;      //   |
    public String [][] ELSEstmt;          //  /

    //Dimension
    public float D_radius = 1;
    public float D_height = 1;
    public float D_x = 1;
    public float D_y = 1;
    public float D_z = 1;
    public boolean D_top = true;
    public boolean D_side = true;
    public boolean D_bottom = true;

    //text
    public String text = " ";

    //position
    public float pos_x = 0.0f;
    public float pos_y = 0.0f;
    public float pos_z = 0.0f;

    //rotation
    public float rot_x = 0;
    public float rot_y = 0;
    public float rot_z = 0;
    public float rot_angle = 0;

    //scale
    public float scale_x = 1;
    public float scale_y = 1;
    public float scale_z = 1;

    //colour
    public float color_R = 1;
    public float color_G = 1;
    public float color_B = 1;

    //keyframes
    private String KFPos = "";
    private String KFRot = "";
    private String KFSca_x = "";
    private String KFSca_y = "";
    private String KFSca_z = "";
    private String KFCol = "";

    //keyvalues
    float KV_Pos_x[] = new float[6];
    float KV_Pos_y[] = new float[6];
    float KV_Pos_z[] = new float[6];
    float KV_Rot_x[] = new float[6];
    float KV_Rot_y[] = new float[6];
    float KV_Rot_z[] = new float[6];
    float KV_Rot_angle[] = new float[6];
    float KV_Scale_x[] = new float[6];
    float KV_Scale_y[] = new float[6];
    float KV_Scale_z[] = new float[6];
    float KV_Color_R[] = new float[6];
    float KV_Color_G[] = new float[6];
    float KV_Color_B[] = new float[6];

    //constructor
    public ObjInfo(mainEditor mainEdit)
    {
        name = "Object" + pos;
        this.parent = mainEdit;
        this.vrmlBuild = mainEdit.vrmlBuild;
        resetValues();
    }

    //set Object1, Object1 etc
    public void setString()
    {
        name = "Object" + pos;
    }

    public String toString()
    {
        return objName;
    }

    //calculate keyvalues for animation
    //updates kv string values for current object too
    public void processAnimation(AniInfo curAni)
    {
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
//if only some animations are enabled, list them in argument
resetValues();

//calculate new values

for (int j=0; j<6; j++)
{
    KV_Pos_x[j] = pos_x + curAni.KV_Pos_x[j];
    KV_Pos_y[j] = pos_y + curAni.KV_Pos_y[j];
    KV_Pos_z[j] = pos_z + curAni.KV_Pos_z[j];
    KV_Rot_x[j] = rot_x + curAni.KV_Rot_x[j];
    KV_Rot_y[j] = rot_y + curAni.KV_Rot_y[j];
    KV_Rot_z[j] = rot_z + curAni.KV_Rot_z[j];
    KV_Rot_angle[j] = rot_angle + curAni.KV_Rot_angle[j];
    KV_Scale_x[j] = scale_x + curAni.KV_Scale_x[j];
    KV_Scale_y[j] = scale_y + curAni.KV_Scale_y[j];
    KV_Scale_z[j] = scale_z + curAni.KV_Scale_z[j];
    KV_Color_R[j] = color_R + curAni.KV_Color_R[j];
    KV_Color_G[j] = color_G + curAni.KV_Color_G[j];
    KV_Color_B[j] = color_B + curAni.KV_Color_B[j];
}

checkValues();

KFPos = String.valueOf(KV_Pos_x[0]) + " " +
        String.valueOf(KV_Pos_y[0]) + " " +
        String.valueOf(KV_Pos_z[0]) + ", " +
        String.valueOf(KV_Pos_x[1]) + " " +
        String.valueOf(KV_Pos_y[1]) + " " +
        String.valueOf(KV_Pos_z[1]) + ", " +
        String.valueOf(KV_Pos_x[2]) + " " +
        String.valueOf(KV_Pos_y[2]) + " " +
        String.valueOf(KV_Pos_z[2]) + ", " +
        String.valueOf(KV_Pos_x[3]) + " " +
        String.valueOf(KV_Pos_y[3]) + " " +
        String.valueOf(KV_Pos_z[3]) + ", " +
        String.valueOf(KV_Pos_x[4]) + " " +
        String.valueOf(KV_Pos_y[4]) + " " +
        String.valueOf(KV_Pos_z[4]) + ", " +
        String.valueOf(KV_Pos_x[5]) + " " +
        String.valueOf(KV_Pos_y[5]) + " " +
        String.valueOf(KV_Pos_z[5]) ;

KFRot = String.valueOf(KV_Rot_x[0]) + " " +
        String.valueOf(KV_Rot_y[0]) + " " +
        String.valueOf(KV_Rot_z[0]) + " " +
        String.valueOf(KV_Rot_angle[0]) + ", " +
        String.valueOf(KV_Rot_x[1]) + " " +
        String.valueOf(KV_Rot_y[1]) + " " +
        String.valueOf(KV_Rot_z[1]) + " " +
        String.valueOf(KV_Rot_angle[1]) + ", " +
        String.valueOf(KV_Rot_x[2]) + " " +
        String.valueOf(KV_Rot_y[2]) + " " +
        String.valueOf(KV_Rot_z[2]) + " " +
        String.valueOf(KV_Rot_angle[2]) + ", " +
        String.valueOf(KV_Rot_x[3]) + " " +
        String.valueOf(KV_Rot_y[3]) + " " +
        String.valueOf(KV_Rot_z[3]) + " " +
        String.valueOf(KV_Rot_angle[3]) + ", " +
        String.valueOf(KV_Rot_x[4]) + " " +
        String.valueOf(KV_Rot_y[4]) + " " +
        String.valueOf(KV_Rot_z[4]) + " " +
        String.valueOf(KV_Rot_angle[4]) + ", " +
        String.valueOf(KV_Rot_x[5]) + " " +
        String.valueOf(KV_Rot_y[5]) + " " +
        String.valueOf(KV_Rot_z[5]) + " " +
        String.valueOf(KV_Rot_angle[5]);

KFSca_x = String.valueOf(KV_Scale_x[0]) + ", " +
        String.valueOf(KV_Scale_x[1]) + ", " +
        String.valueOf(KV_Scale_x[2]) + ", " +
        String.valueOf(KV_Scale_x[3]) + ", " +
        String.valueOf(KV_Scale_x[4]) + ", " +
        String.valueOf(KV_Scale_x[5]) ;

KFSca_y = String.valueOf(KV_Scale_y[0]) + ", " +
        String.valueOf(KV_Scale_y[1]) + ", " +
        String.valueOf(KV_Scale_y[2]) + ", " +
        String.valueOf(KV_Scale_y[3]) + ", " +
        String.valueOf(KV_Scale_y[4]) + ", " +
        String.valueOf(KV_Scale_y[5]) ;

KFSca_z = String.valueOf(KV_Scale_z[0]) + ", " +
        String.valueOf(KV_Scale_z[1]) + ", " +
        String.valueOf(KV_Scale_z[2]) + ", " +
        String.valueOf(KV_Scale_z[3]) + ", " +
        String.valueOf(KV_Scale_z[4]) + ", " +
        String.valueOf(KV_Scale_z[5]) ;

KFCol = String.valueOf(KV_Color_R[0]) + " " +
        String.valueOf(KV_Color_G[0]) + " " +
        String.valueOf(KV_Color_B[0]) + ", " +
        String.valueOf(KV_Color_R[1]) + " " +
        String.valueOf(KV_Color_G[1]) + " " +
        String.valueOf(KV_Color_B[1]) + ", " +
        String.valueOf(KV_Color_R[2]) + " " +
        String.valueOf(KV_Color_G[2]) + " " +
        String.valueOf(KV_Color_B[2]) + ", " +
        String.valueOf(KV_Color_R[3]) + " " +
        String.valueOf(KV_Color_G[3]) + " " +
        String.valueOf(KV_Color_B[3]) + ", " +
        String.valueOf(KV_Color_R[4]) + " " +
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
            String.valueOf(KV_Color_G[4]) + " " +
            String.valueOf(KV_Color_B[4]) + ", " +
            String.valueOf(KV_Color_R[5]) + " " +
            String.valueOf(KV_Color_G[5]) + " " +
            String.valueOf(KV_Color_B[5]) ;
}

//updates static values for pos, color, rot
public void updateChanges()
{
    int field_type;
    boolean result;
    String Field;
    String Pos, Rot, Sca, Col;

    Pos = pos_x + " " + pos_y + " " + pos_z;
    Rot = rot_x + " " + rot_y + " " + rot_z + " " + rot_angle;
    Sca = scale_x + " " + scale_y + " " + scale_z;
    Col = color_R + " " + color_G + " " + color_B;

    VRMLObject vrmlObj = this.vrmlObj;

    EventInSFVec3f Translation = vrmlObj.Translation;
    EventInSFRotation Rotation = vrmlObj.Rotation;
    EventInSFVec3f Scale = vrmlObj.Scale;
    EventInSFColor Color = vrmlObj.Color;

    vrml.external.Node transform = vrmlObj.transform;
    vrml.external.Node material = vrmlObj.material[0];

    //update shape
    //remove entire shape node and add new one
    vrmlObj.valuesChanged();

    //update position
    Field = "translation";
    field_type = vrmlBuild.getFieldType(transform,Field);
    result = vrmlBuild.changeEAI(transform, Field, new MFValue(field_type, Pos));

    //update orientation
    Field = "rotation";
    field_type = vrmlBuild.getFieldType(transform,Field);
    result = vrmlBuild.changeEAI(transform, Field, new MFValue(field_type, Rot));

    //update color
    Field = "diffuseColor";
    field_type = vrmlBuild.getFieldType(material,Field);
    result = vrmlBuild.changeEAI(material, Field, new MFValue(field_type, Col));
}

//this function is for getting the latest position, color, rotation values
public void getLatestValues()
{
    StringTokenizer st;

    vrmlObj.getState();

    //update position
    st = new StringTokenizer(vrmlObj.ValueTrans);
    pos_x = Float.valueOf(st.nextToken()).floatValue();
    pos_y = Float.valueOf(st.nextToken()).floatValue();
    pos_z = Float.valueOf(st.nextToken()).floatValue();

    //update rotation
    st = new StringTokenizer(vrmlObj.ValueRot);
    rot_x = Float.valueOf(st.nextToken()).floatValue();
    rot_y = Float.valueOf(st.nextToken()).floatValue();
    rot_z = Float.valueOf(st.nextToken()).floatValue();
    rot_angle = Float.valueOf(st.nextToken()).floatValue();

    //update color
    st = new StringTokenizer(vrmlObj.ValueColor);
    color_R = Float.valueOf(st.nextToken()).floatValue();
    color_G = Float.valueOf(st.nextToken()).floatValue();
    color_B = Float.valueOf(st.nextToken()).floatValue();

}

//updates the interpolator values of corresponding vrml object
public void previewAnimations(VRMLBuilder vrmlBuild)
{
    int field_type;
    boolean result;
    String Field;

    vrml.external.Node Position_Interpolator = vrmlObj.Position_Interpolator;
    vrml.external.Node Orientation_Interpolator = vrmlObj.Orientation_Interpolator;
    vrml.external.Node Scalar_Interpolator = vrmlObj.Scalar_Interpolator;
    vrml.external.Node Color_Interpolator = vrmlObj.Color_Interpolator;

    //animations
    Field = "keyValue";

    field_type = vrmlBuild.getFieldType(Position_Interpolator,Field);
    result = vrmlBuild.changeEAI(Position_Interpolator, Field, new MFValue(field_type, KFPos));

    field_type = vrmlBuild.getFieldType(Orientation_Interpolator,Field);
    result = vrmlBuild.changeEAI(Orientation_Interpolator, Field, new MFValue(field_type, KFRot));

    field_type = vrmlBuild.getFieldType(Scalar_Interpolator,Field);
    result = vrmlBuild.changeEAI(Scalar_Interpolator, Field, new MFValue(field_type, KFSca_x));
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
    field_type = vrmlBuild.getFieldType(Color_Interpolator,Field);
    result = vrmlBuild.changeEAI(Color_Interpolator, Field, new MFValue(field_type, KFCol));

    vrmlObj.startAnimation(vrmlBuild);
}

//this function checks for illegal value for scale and color
public void checkValues()
{
    for (int m=0; m <= 5; m++)
    {
        if (KV_Scale_x[m] == 0)
        {
            KV_Scale_x[m] = Float.valueOf("0.1").floatValue();
        }
        if (KV_Scale_y[m] == 0) {
            KV_Scale_y[m] = Float.valueOf("0.1").floatValue();
        }
        if (KV_Scale_z[m] == 0) {
            KV_Scale_z[m] = Float.valueOf("0.1").floatValue();
        }
        if (KV_Color_R[m] > 1.0)
        {
            KV_Color_R[m] = Float.valueOf("1.0").floatValue();
        } else if (KV_Color_R[m] < 0.0)
        {
            KV_Color_R[m] = Float.valueOf("0.0").floatValue();
        }
        if (KV_Color_G[m] > 1.0)
        {
            KV_Color_G[m] = Float.valueOf("1.0").floatValue();
        } else if (KV_Color_G[m] < 0.0)
        {
            KV_Color_G[m] = Float.valueOf("0.0").floatValue();
        }
        if (KV_Color_B[m] > 1.0)
        {
            KV_Color_B[m] = Float.valueOf("1.0").floatValue();
        } else if (KV_Color_B[m] < 0.0)
        {
            KV_Color_B[m] = Float.valueOf("0.0").floatValue();
        }
    }
}

//this function resets the keyvalues to default values
public void resetValues()
{
    //reset values
    for (int j=0; j<6; j++) {
        KV_Pos_x[j] = 0;
        KV_Pos_y[j] = 0;
        KV_Pos_z[j] = 0;
        KV_Rot_x[j] = 0;
        KV_Rot_y[j] = 0;
        KV_Rot_z[j] = 0;
        KV_Rot_angle[j] = 0;
        KV_Scale_x[j] = 1;
        KV_Scale_y[j] = 1;
        KV_Scale_z[j] = 1;
        KV_Color_R[j] = 0;
        KV_Color_G[j] = 0;
        KV_Color_B[j] = 0;
    }
}

//this function is called by vrmlObject's callback
//it processes the script
public void isClicked()
{
    boolean condMet = false; //initialise to false
    int i;

    System.out.println(this.objName + " is clicked :'D");

    if (IFexp.length != 0)
    {
        condMet = processExp(IFexp);
        System.out.println("condMet for IF " + condMet);
        if (condMet == true)
        {
            processStmt(IFstmt);
        } else //condition(s) not met
        {
            //process ELSEIF
            if (ELSEIFexp.length != 0)
            {
                i = 0;
                while (i < ELSEIFexp.length)
                {
                    condMet = processExp(ELSEIFexp[i]);
                    System.out.println("condMet for ELSEIF" + i + " " + condMet);
                    if (condMet == true)
                    {
                        //if condition(s) already met, dun need to process other
                        //else-ifs
                        processStmt(ELSEIFstmt[i]);
                        i = ELSEIFexp.length;
                    } else
                    {
                        //condition(s) not met
                        //continue processing next set
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                        i++;
                    }

                }
            }

            // after processing ELSEIF, check if any condition(s) met
            // if no, then check if else statements exists
            // and process else statements
            if (condMet == false)
            {
                if (ELSEstmt.length != 0)
                {
                    processStmt(ELSEstmt);
                }
            }
        }

    } else
    {
        //empty script
        System.out.println("empty script");
    }

}

//process expressions
public boolean processExp(String [][] stmt)
{
    ObjInfo tmpObjInfo; // Object
    VRMLObject tmpVrmlObj; //vrml object
    String ppty; //property
    String value; //value
    boolean result = false;

    if (stmt.length == 1)
    {
        //1 expression
        tmpObjInfo = (ObjInfo) parent.ObjectTable.get(stmt[0][0].trim());
        tmpVrmlObj = tmpObjInfo.vrmlObj;
        tmpVrmlObj.getState();
        ppty = stmt[0][1].trim();
        value = stmt[0][2].trim();

        result = compareValues(ppty, value, tmpVrmlObj);

    } else if (stmt.length == 2)
    {
        //2 expressions
        //find out if it's AND/OR
        if (stmt[0][3].trim().equals("&&"))
        {
            //both conditions must be fulfilled.
            //check one first, if false, return immediately.
            //else check second one

            //check first expression
            tmpObjInfo = (ObjInfo) parent.ObjectTable.get(stmt[0][0].trim());
            tmpVrmlObj = tmpObjInfo.vrmlObj;
            tmpVrmlObj.getState();
            ppty = stmt[0][1].trim();
            value = stmt[0][2].trim();

            result = compareValues(ppty, value, tmpVrmlObj);

            if (result == false)
            {
                return result;
            } else
            {
                //check second expression
                tmpObjInfo = (ObjInfo) parent.ObjectTable.get(stmt[1][0].trim());
                tmpVrmlObj = tmpObjInfo.vrmlObj;
                tmpVrmlObj.getState();
                ppty = stmt[1][1].trim();
                value = stmt[1][2].trim();

                result = compareValues(ppty, value, tmpVrmlObj);
                // T&&T = T
                // T&&F = F
                // final outcome is directly dependent on second expression
                return result;
            }
        } else
        {
            //if one true, return immediately
            //else check second

            //check first expression
            tmpObjInfo = (ObjInfo) parent.ObjectTable.get(stmt[0][0].trim());
            tmpVrmlObj = tmpObjInfo.vrmlObj;
            tmpVrmlObj.getState();
            ppty = stmt[0][1].trim();
            value = stmt[0][2].trim();

            result = compareValues(ppty, value, tmpVrmlObj);

            if (result == true)
            {
                return result;
            } else
            {
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                //check second expression
                tmpObjInfo = (ObjInfo) parent.ObjectTable.get(stmt[1][0].trim());
                tmpVrmlObj = tmpObjInfo.vrmlObj;
                tmpVrmlObj.getState();
                ppty = stmt[1][1].trim();
                value = stmt[1][2].trim();

                result = compareValues(ppty, value, tmpVrmlObj);
                // F||T = T
                // F||F = F
                // final outcome is directly dependent on second expression
                return result;
            }
        }
    }
    return result;
}

//process statments
public void processStmt(String [][] stmt)
{
    ObjInfo tmpObjInfo; // Object
    AniInfo tmpAniInfo; //animation
    StringTokenizer tmptok;
    int i = 0;

    for (int j = 0; j < stmt.length; j++)
    {
        tmpObjInfo = (ObjInfo) parent.ObjectTable.get(stmt[j][0].trim());

        //update latest values
        tmpObjInfo.getLatestValues();

        tmptok = new StringTokenizer(stmt[j][2]);

        //reset i
        i = 0;

        if (stmt[j][1].trim().toLowerCase().equals("rotation"))
        {
            while (tmptok.hasMoreTokens())
            {
                if (i == 0 )
                {
                    tmpObjInfo.rot_x = Float.valueOf(tmptok.nextToken()).floatValue();
                } else if (i == 1)
                {
                    tmpObjInfo.rot_y = Float.valueOf(tmptok.nextToken()).floatValue();
                } else if (i == 2)
                {
                    tmpObjInfo.rot_z = Float.valueOf(tmptok.nextToken()).floatValue();
                } else
                {
                    tmpObjInfo.rot_angle = Float.valueOf(tmptok.nextToken()).floatValue();
                }
                i++;
            }
            tmpObjInfo.updateChanges();
        } else if (stmt[j][1].trim().toLowerCase().equals("position"))
        {
            while (tmptok.hasMoreTokens())
            {

                if (i == 0 )
                {
                    tmpObjInfo.pos_x = Float.valueOf(tmptok.nextToken()).floatValue();
                } else if (i == 1)
                {
                    tmpObjInfo.pos_y = Float.valueOf(tmptok.nextToken()).floatValue();
                } else
                {
                    tmpObjInfo.pos_z = Float.valueOf(tmptok.nextToken()).floatValue();
                }
                i++;
            }
            tmpObjInfo.updateChanges();
        } else if (stmt[j][1].trim().toLowerCase().equals("color"))
        {
            while (tmptok.hasMoreTokens())
            {

                if (i == 0 )
                {
                    tmpObjInfo.color_R = Float.valueOf(tmptok.nextToken()).floatValue();
                } else if (i == 1)
                {
                    tmpObjInfo.color_G = Float.valueOf(tmptok.nextToken()).floatValue();
                } else
                {
                    tmpObjInfo.color_B = Float.valueOf(tmptok.nextToken()).floatValue();
                }
                i++;
            }
            tmpObjInfo.updateChanges();
        } else
        {
            //animation
            if (stmt[j][2].toString().trim().toLowerCase().equals("true"))
            {
                tmpAniInfo = (AniInfo) parent.AniChoiceTable.get(stmt[j][1].trim());
                System.out.println("Animation " + tmpAniInfo.aniName);
                tmpObjInfo.processAnimation(tmpAniInfo);
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                    tmpObjInfo.previewAnimations(parent.vrmlBuild);
                } else
                {
                    //stop animation
                }
            }
        }
    }

    //compare values indicated in conditions
    private boolean compareValues(String property, String defined, VRMLObject vrmlObject)
    {
        // property can be position, rotation, or color
        //defined is the values defined by user
        //vrmlObject is the object of interest

        StringTokenizer st; //stores user-entered value
        StringTokenizer st1; //stores value from getState
        float val; //stores tmp float value from user
        float val1; //stores tmp float value from getStare

        if (property.trim().equals("position"))
        {
            st = new StringTokenizer(defined.trim());
            st1 = new StringTokenizer(vrmlObject.ValueTrans.trim());

            while (st.hasMoreTokens())
            {
                val = Float.valueOf(st.nextToken()).floatValue();
                val1 = Float.valueOf(st1.nextToken()).floatValue();
                if (val != val1)
                {
                    return false;
                }
            }

        } else if (property.trim().equals("rotation"))
        {
            st = new StringTokenizer(defined.trim());
            st1 = new StringTokenizer(vrmlObject.ValueRot.trim());

            while (st.hasMoreTokens())
            {
                val = Float.valueOf(st.nextToken()).floatValue();
                val1 = Float.valueOf(st1.nextToken()).floatValue();
                if (val != val1)
                {
                    return false;
                }
            }

        } else if (property.trim().equals("color"))
        {
            st = new StringTokenizer(defined.trim());
            st1 = new StringTokenizer(vrmlObject.ValueColor.trim());

            while (st.hasMoreTokens())
            {
                val = Float.valueOf(st.nextToken()).floatValue();
                val1 = Float.valueOf(st1.nextToken()).floatValue();
                if (val != val1)
                {
                    return false;
                }
            }
        }
        return true;
    }
}
```

//--------------scriptParser.java--------------//

```
import java.lang.String;
import java.lang.StringBuffer;
import java.util.*;

public class scriptParser
{
    String [][] IFexp;
    String [][] IFstmt;
    String [][][] ELSEIFexp;
    String [][][] ELSEIFstmt;
    String [][] ELSEstmt;

    ObjInfo curObj;

    //constructor
    public scriptParser(ObjInfo objInfo)
    {
        this.curObj = objInfo;
    }

    //break up the chunk into token blocks
    public String [] getTokens(String original)
    {
        String parseline = original;
        StringBuffer bf = new StringBuffer(parseline);
        StringTokenizer st;
        String [] result = new String [0];

        try
        {
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
        int position = parseline.indexOf("(");  //for conditions
        while (position != -1)
        {
            bf = bf.insert(position, " ");
            bf = bf.insert(position+2, " ");
            parseline = bf.toString();
            position = parseline.indexOf("(", position+2);
        }

        position = parseline.indexOf(")");//for conditions
        while (position != -1)
        {
            bf = bf.insert(position, " ");
            bf = bf.insert(position+2, " ");
            parseline = bf.toString();
            position = parseline.indexOf(")", position+2);
        }

        position = parseline.indexOf("{");  //for conditions
        while (position != -1)
        {
            bf = bf.insert(position, " ");
            bf = bf.insert(position+2, " ");
            parseline = bf.toString();
            position = parseline.indexOf("{", position+2);
        }

        position = parseline.indexOf("}");//for conditions
        while (position != -1)
        {
            bf = bf.insert(position, " ");
            bf = bf.insert(position+2, " ");
            parseline = bf.toString();
            position = parseline.indexOf("}", position+2);
        }

        position = parseline.indexOf("=");//for equal sign
        while (position != -1)
        {
            bf = bf.insert(position, " ");
            bf = bf.insert(position+2, " ");
            parseline = bf.toString();
            position = parseline.indexOf("=", position+2);
        }

        position = parseline.indexOf("!");//for not sign
        while (position != -1)
        {
            bf = bf.insert(position, " ");
            bf = bf.insert(position+2, " ");
            parseline = bf.toString();
            position = parseline.indexOf("!", position+2);
        }

        //      the tokens will be in this form
        //      tok1 --> IF ( expression )
        //      tok2 --> statements ; statements ;
        //      tok3 --> ELSEIF ( expression )
        //      tok3 --> statements ; statements ;
        //      tok3 --> ELSE ( expression )
        //      tok3 --> statements ; statements ;
        //      tok3 --> ENDIF
        st = new StringTokenizer(parseline, "{}");
        String [] tmp = new String [st.countTokens()];

        int i =0;
        while (st.hasMoreTokens())
        {
            tmp[i] = st.nextToken();
            i++ ;
        }
        result = tmp;
    }
    catch (Exception e)
    {
        System.out.print("error in script parser try loop");
        System.out.print(e.toString());
    }
    finally
    {
        return result;
    }
}

private String [] TokenizeDot(String original)
{
    String parseline = original;
    StringBuffer bf = new StringBuffer(parseline);
    StringTokenizer st;
    String [] result;
    int i = 0;

    int position = parseline.indexOf(".");//for state
    while (position != -1)
    {
        bf = bf.insert(position, " ");
        bf = bf.insert(position+2, " ");
        parseline = bf.toString();
        position = parseline.indexOf(".", position+2);
    }

    st = new StringTokenizer(parseline, ".");
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
        result = new String [st.countTokens()];
        while (st.hasMoreTokens())
        {
            result[i] = st.nextToken().trim();
            i++ ;

        }
        return result;
    }

    public boolean processTokens(String[] tokens)
    {
        String curTok;
        int i = 0; //big loop
        int j = 0;
        int k = 0; //small expression (no. of words in one stmt)
        int m = 0, n = 0; //2-dimensional array
        int a = 0; int b = 0; int c = 0; //3-dimensional array
        int cnt = 0;
        int cntStmt = 0; //store the number of statements
        int cntELSEIF = 0; //store the number of ELSEIFs
        boolean result = true;
        StringTokenizer tok, tok1;
        String [] tmp;
        String [] tmp1;
        String [] tmp2;
        String [][] exp = new String [2][];

        if (tokens.length == 0)
        {
            System.out.println("empty script");
            result = false;
        }

        for (i = 0; i < tokens.length; i++)
        {
            if ( tokens[i].trim().toUpperCase().startsWith("ELSEIF"))
            {
                cntELSEIF++;
            }
        }
        ELSEIFexp = new String[cntELSEIF][][];
        ELSEIFstmt = new String[cntELSEIF][][];

        //      check for syntax errors
        //      the tokens will be in this form
        //      tok1 --> IF ( expression )
        //      tok2 --> statements ; statements ;
        //      tok3 --> ELSEIF ( expression )
        //      tok3 --> statements ; statements ;
        //      tok3 --> ELSE ( expression )
        //      tok3 --> statements ; statements ;
        //      tok3 --> ENDIF

        for (i=0; i < tokens.length; i++)
        {
            curTok = tokens[i];
            //remember each token is a string that may contain more than one term
            if (i == 0)
            {
                //      first token
                //      further break up into more tokens and analyse one by one
                //      should get this
                //      first token --> IF
                //      first token --> (
                //      first token --> expression
                //      first token --> )

                tok = new StringTokenizer(curTok.trim(), "(=) ");
                cnt = tok.countTokens();
                tmp = new String [cnt];
                while (tok.hasMoreTokens())
                {
                    tmp[m] = tok.nextToken().trim();
                    m++;
                }

                if (cnt == 3)
                {
                    //one expression
                    IFexp = new String[1][3];
                    for ( k = 0; k < 3 ; k++)
                    {
                        if (k == 1)
                        {
                            // EG: object.position
                            tmp1 = TokenizeDot(tmp[k]);

                            IFexp[0][n] = tmp1[0];
                            n++ ;
                            IFexp[0][n] = tmp1[1];
                            n++;
                        } else if (k == 2)
                        {
                            IFexp[0][n] = new StringTokenizer(tmp[k], "'").nextToken().trim().replace(','
, ' ');                                 // EG: '0.5 0.5 0.5'
                            n++;
                        }
                    }

                } else if (cnt == 6)
                {
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                    IFexp = new String[2][4];
                    n = 0; //reset n
                    for ( k = 0; k < 4 ; k++)
                    {
                        if (k == 1)
                        {
                            // EG: object.position
                            tmp1 = TokenizeDot(tmp[k]);

                            IFexp[0][n] = tmp1[0];
                            n++ ;
                            IFexp[0][n] = tmp1[1];
                            n++;
                        } else if (k == 2)
                        {
                            IFexp[0][n] = new StringTokenizer(tmp[k], "'").nextToken().trim().replace(','
, ' ');                                 // EG: '0.5 0.5 0.5'
                            n++;
                        } else if ( k == 3)
                        {
                            IFexp[0][n] = tmp[k]; // && or ||
                            n++;
                        }
                    }

                    n = 0; //reset n
                    for (k = 4; k < 6; k ++)
                    {
                        if (k == 4)
                        {
                            // EG: object.position
                            tmp1 = TokenizeDot(tmp[k]);

                            IFexp[1][n] = tmp1[0];
                            n++ ;
                            IFexp[1][n] = tmp1[1];
                            n++;
                        } else if (k == 5)
                        {
                            IFexp[1][n] = new StringTokenizer(tmp[k],"'").nextToken().trim().replace(',' ,
' ');
                            // EG: '0.5 0.5 0.5'
                            n++;
                        }
                    }
                } else
                {
                    System.out.println("syntax error in IF expression. please check.");
                }

                //process statements
                curTok = tokens[i+1];
                tok = new StringTokenizer(curTok.trim(), ";");
                cntStmt = tok.countTokens();
                tmp = new String [cntStmt];
                m = 0;
                while (tok.hasMoreTokens())
                {
                    tmp[m] = tok.nextToken();    //object.position='1,2,0'
                    m++;
                }

                //loop through each statement and collect the information
                IFstmt = new String[cntStmt][3];
                for (m = 0; m < cntStmt; m++)
                {
                    tok1 = new StringTokenizer(tmp[m],"=");
                    n = 0;
                    tmp1 = new String[tok1.countTokens()];
                    while (tok1.hasMoreTokens())
                    {
                        tmp1[n] = tok1.nextToken().trim();
                        n++;
                    }

                    tmp2 = TokenizeDot(tmp1[0]);

                    n = 0;
                    IFstmt[m][n] = tmp2[0];
                    n++;
                    IFstmt[m][n] = tmp2[1];
                    n++;
                    IFstmt[m][n] = new StringTokenizer(tmp1[1],"'").nextToken().trim().replace(',' , ' ');

                }
            }
            else if( i == tokens.length - 1) // i == tokens.length - 1
            {
                // last token
                //should get this
                //      last token --> ENDIF

                if (curTok.trim().toUpperCase().compareTo("ENDIF") != 0)
                {
                    //syntax error
                    System.out.println("ERROR - script should end with 'ENDIF'");
                }
            } else
            {
                if (curTok.trim().toUpperCase().startsWith("ELSEIF"))
                {
                    //process expressions
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                    tok = new StringTokenizer(curTok.trim(), "(=) ");
                    cnt = tok.countTokens();
                    tmp = new String [cnt];
                    m = 0;
                    n = 0;
                    while (tok.hasMoreTokens())
                    {
                        tmp[m] = tok.nextToken().trim();
                        m++;
                    }

                    if (cnt == 3)
                    {
                        //one expression
                        ELSEIFexp[a] = new String [1][3];
                        for ( k = 0; k < 3 ; k++)
                        {
                            if (k == 1)
                            {
                                // EG: object.position
                                tmp1 = TokenizeDot(tmp[k]);

                                ELSEIFexp[a][0][n] = tmp1[0];
                                n++ ;
                                ELSEIFexp[a][0][n] = tmp1[1];
                                n++;
                            } else if (k == 2)
                            {
                                ELSEIFexp[a][0][n] =
                                        new StringTokenizer(tmp[k], "'").nextToken().trim().replace(',' ,
' ');
                                // EG: '0.5 0.5 0.5'
                                n++;
                            }
                        }


                    } else if (cnt == 6)
                    {
                        ELSEIFexp[a] = new String[2][4];
                        n = 0; //reset n
                        for ( k = 0; k < 4 ; k++)
                        {
                            if (k == 1)
                            {
                                // EG: object.position
                                tmp1 = TokenizeDot(tmp[k]);

                                ELSEIFexp[a][0][n] = tmp1[0];
                                n++ ;
                                ELSEIFexp[a][0][n] = tmp1[1];
                                n++;
                            } else if (k == 2)
                            {
                                ELSEIFexp[a][0][n] =
                                        new StringTokenizer(tmp[k],"'").nextToken().trim().replace(',' , '
');
                                // EG: '0.5 0.5 0.5'
                                n++;
                            } else if ( k == 3)
                            {
                                ELSEIFexp[a][0][n] = tmp[k]; // && or ||
                                n++;
                            }
                        }

                        n = 0; //reset n
                        for (k = 4; k < 6; k ++)
                        {
                            if (k == 4)
                            {
                                // EG: object.position
                                tmp1 = TokenizeDot(tmp[k]);

                                ELSEIFexp[a][1][n] = tmp1[0];
                                n++ ;
                                ELSEIFexp[a][1][n] = tmp1[1];
                                n++;
                            } else if (k == 5)
                            {
                                ELSEIFexp[a][1][n] =
                                        new StringTokenizer(tmp[k],"'").nextToken().trim().replace(',' , '
');
                                // EG: '0.5 0.5 0.5'
                                n++;
                            }
                        }

                    } else
                    {
                        System.out.println("syntax in ELSEIF expression! please check");
                    }

                    //process statements
                    curTok = tokens[i+1];
                    tok = new StringTokenizer(curTok.trim(), ";");
                    cntStmt = tok.countTokens();
                    tmp = new String [cntStmt];
                    m = 0;
                    while (tok.hasMoreTokens())
                    {
                        tmp[m] = tok.nextToken();    //object.position='1,2,0'
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                    m++;
                }

                //loop through each statement and collect the information
                ELSEIFstmt[a] = new String[cntStmt][3];
                for (m = 0; m < cntStmt; m++)
                {
                    tok1 = new StringTokenizer(tmp[m],"=");
                    n = 0;
                    tmp1 = new String[tok1.countTokens()];
                    while (tok1.hasMoreTokens())
                    {
                        tmp1[n] = tok1.nextToken().trim();
                        n++;
                    }

                    tmp2 = TokenizeDot(tmp1[0]);

                    n = 0;
                    ELSEIFstmt[a][m][n] = tmp2[0];
                    n++;
                    ELSEIFstmt[a][m][n] = tmp2[1];
                    n++;
                    ELSEIFstmt[a][m][n] =
                                new StringTokenizer(tmp1[1],"'").nextToken().trim().replace(',' , '
');
                }
                a++;
            } else if (curTok.trim().toUpperCase().startsWith("ELSE"))
            {
                //process statements
                curTok = tokens[i+1];
                tok = new StringTokenizer(curTok.trim(), ";");
                cntStmt = tok.countTokens();
                tmp = new String [cntStmt];
                m = 0;
                while (tok.hasMoreTokens())
                {
                    tmp[m] = tok.nextToken();    //object.position='1,2,0'
                    m++;
                }

                //loop through each statement and collect the information

                ELSEstmt = new String[cntStmt][3];
                for (m = 0; m < cntStmt; m++)
                {
                    tok1 = new StringTokenizer(tmp[m],"=");
                    n = 0;
                    tmp1 = new String[tok1.countTokens()];
                    while (tok1.hasMoreTokens())
                    {
                        tmp1[n] = tok1.nextToken().trim();
                        n++;
                    }

                    tmp2 = TokenizeDot(tmp1[0]);

                    n = 0;
                    ELSEstmt[m][n] = tmp2[0];
                    n++;
                    ELSEstmt[m][n] = tmp2[1];
                    n++;
                    ELSEstmt[m][n] = new StringTokenizer(tmp1[1],"'").nextToken().trim().replace(',' ,
' ');
                }
            }

            //should get these
            //  token0 --> statement1
            //  token0 --> ;
            //  token0 --> statement2
            //  token0 --> ;
            //  token1 --> ELSEIF
            //  token1 --> (
            //  token1 --> expression
            //  token1 --> )
            //  token2 --> statement1
            //  token2 --> ;
            //  token2 --> statement2
            //  token2 --> ;
            //  token3 --> ELSE
            //  token4 --> statement1
            //  token4 --> ;
        }
    }
    return result;
  }
}
```

//--------------VRMLBuilder.java--------------//

```
//this class parses through the jtree and generates the VRML scene

//maths parser api package
import org.nfunk.jep.function.*;
import org.nfunk.jep.*;

//java classes for reading and writing
import java.io.*;

//jtree
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
import javax.swing.*;
import javax.swing.tree.*;

//others
import java.util.*;
import java.lang.Float;
import vrml.external.field.*;
import vrml.external.exception.*;
import vrml.external.Node;
import vrml.external.Browser;
import vrml.field.*;
import vrml.node.*;
import com.ms.security.*;
import com.ms.security.permissions.*;

//for writing into user's system
import java.security.*;

public class VRMLBuilder extends Object implements EventOutObserver
{
    // Browser we're using
    Browser browser;

    // Root of the scene graph (to which we add our nodes)
    vrml.external.Node Scene;

    //JEP Object
    JEP jep = new JEP();

    //constructor
    public VRMLBuilder (Browser browser, vrml.external.Node Scene)
    {
        this.browser = browser;
        this.Scene = Scene;

        //add standard constants such as "pi" and "e"
        jep.addStandardConstants();
        //add standard functions such as trigonometric expressions
        jep.addStandardFunctions();
    }

    public VRMLObject VRMLGen(ObjInfo objNew)
    {
        VRMLObject vrmlNew = new VRMLObject(browser, objNew);
        EventInMFNode addChildren = (EventInMFNode)Scene.getEventIn("addChildren");
        addChildren.setValue(vrmlNew.Object_array);
        objNew.vrmlObj = vrmlNew;
        return vrmlNew;
    }

    public void removeObject(ObjInfo theObj)
    {
        VRMLObject theObject = theObj.vrmlObj;
        EventInMFNode removeChildren = (EventInMFNode) Scene.getEventIn("removeChildren");
        removeChildren.setValue(theObject.Object_array);
    }

    public void removeAll(Hashtable objectsTable)
    {
        Enumeration keys = objectsTable.elements();
        while (keys.hasMoreElements())
        {
            ObjInfo objInfo = (ObjInfo)keys.nextElement();
            System.out.println(objInfo.objName + " is removed from the scene");
            VRMLObject vrmlObject = objInfo.vrmlObj;
            EventInMFNode removeChildren = (EventInMFNode) Scene.getEventIn("removeChildren");
            removeChildren.setValue(vrmlObject.Object_array);
        }
    }

    /**
     * The method to create this VRML Object to one location.
     * @param   String newToLocation    new location to save this VRML Object
     * @param   boolean NeedPolicy      The flag indicate whether needing permission to
     *                                  creating these texture files.
     * @return  boolean
     */
    public boolean createVRMLFile(String newToLocation, String fn, boolean NeedPolicy)
    {
        try
        {
            if (NeedPolicy==true)
            {
                //PolicyEngine.assertPermission(PermissionID.FILEIO);
            }

            File f;
            if (!newToLocation.equals(""))
            {
                f = new File(newToLocation+fn);
            }
            else
            {
                f = new File(newToLocation+fn);
            }

            if (f != null)
            {
                FileOutputStream outStream = new FileOutputStream(f);
                outStream.write("test".getBytes());
                outStream.close();
                return true;
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
        }
        else
        {
            return false;
        }
    }
    catch(IOException e)
    {
        System.out.println("Cannot create file");
        return false;
    }
}

public void callback(EventOut event, double time, Object userData)
{

}

public boolean changeEAI(vrml.external.Node node, String Field, MFValue value)
{
    try
    {
        vrml.external.Node target_node = node;
        if ( target_node != null )
        {
            // Get the  Field EventIn
            String set_Field = "set_"+Field.trim();//eg.set_type

            //check field type & invoke its EAI method
            EventIn Event_In = target_node.getEventIn(set_Field);

            int field_type = Event_In.getType();
            switch (field_type)
            {
            case FieldTypes.SFBOOL://1
                boolean bool = value.Bool;
                ((EventInSFBool)Event_In).setValue(bool);
                break;

            case FieldTypes.SFIMAGE://2
                /***can't handle now**
                Image image = (Image)value.getImageValue();
                ((EventInSFImage)Event_In).setValue(width, height, numComponets, pixels);*/
                break;

            case FieldTypes.SFTIME://3
                double time = value.Time;
                ((EventInSFTime)Event_In).setValue(time);
                break;

            case FieldTypes.SFCOLOR://4
                float[] color = value.Float_Array;
                ((EventInSFColor)Event_In).setValue(color);
                break;

            case FieldTypes.MFCOLOR://5
                float[][] mfcolor = value.MFloat_Array;
                ((EventInMFColor)Event_In).setValue(mfcolor);
                break;

            case FieldTypes.SFFLOAT://6
                float float_value = value.SFloat;
                ((EventInSFFloat)Event_In).setValue(float_value);
                break;

            case FieldTypes.MFFLOAT://7
                float[] mffloat_value = value.Float_Array;
                ((EventInMFFloat)Event_In).setValue(mffloat_value);
                break;

            case FieldTypes.SFINT32://8
                int int_value = value.SInt;
                ((EventInSFInt32)Event_In).setValue(int_value);
                break;

            case FieldTypes.MFINT32://9
                int[] mfint_value = value.MInt;
                ((EventInMFInt32)Event_In).setValue(mfint_value);
                break;

            case FieldTypes.SFNODE://10
                vrml.external.Node snode = value.SNode;
                ((EventInSFNode)Event_In).setValue(snode);
                break;

            case FieldTypes.MFNODE://11
                vrml.external.Node[] mnode = value.MNode;
                ((EventInMFNode)Event_In).setValue(mnode);
                break;

            case FieldTypes.SFROTATION://12
                float[] rot_value = value.Float_Array;
                ((EventInSFRotation)Event_In).setValue(rot_value);
                break;

            case FieldTypes.MFROTATION://13
                float[][] mrot_value = value.MFloat_Array;
                ((EventInMFRotation)Event_In).setValue(mrot_value);
                break;

            case FieldTypes.SFSTRING://14
                String str = value.SString;
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
                    ((EventInSFString)Event_In).setValue(str);
                    break;

                case FieldTypes.MFSTRING://15
                    String[] mstr = value.MString;
                    ((EventInMFString)Event_In).setValue(mstr);
                    break;

                case FieldTypes.SFVEC2F://16
                    float[] xy = value.Float_Array;
                    ((EventInSFVec2f)Event_In).setValue(xy);
                    break;

                case FieldTypes.MFVEC2F://17
                    float[][] mxy = value.MFloat_Array;
                    ((EventInMFVec2f)Event_In).setValue(mxy);
                    break;

                case FieldTypes.SFVEC3F://18
                    float[] xyz = value.Float_Array;
                    ((EventInSFVec3f)Event_In).setValue(xyz);
                    break;

                case FieldTypes.MFVEC3F://19
                    float[][] mxyz = value.MFloat_Array;
                    ((EventInMFVec3f)Event_In).setValue(mxyz);
                    break;
                }
                //clear up
                target_node = null;
                return true;
            }

            System.out.println("Target node does not exist!!\n");
            return false;
        }
        catch (InvalidEventInException ee)
        {
            System.out.println("Failed to get EventIn:" + ee+"\n");
            return false;
        }
        catch (InvalidEventOutException ee)
        {
            System.out.println("Failed to get EventOut:" + ee+"\n");
            return false;
        }
    }

    public int getFieldType(vrml.external.Node node, String Field)
    {
        // Get the  Field EventIn
        String set_Field = "set_"+Field.trim();//eg.set_type
        //System.out.println("check set field="+set_Field);
        //check field type
        EventIn Event_In = node.getEventIn(set_Field);
        return Event_In.getType();
    }
}
```

//--------------VRMLObject.java--------------//

```
import java.util.*;
import java.io.*;
import javax.swing.tree.DefaultMutableTreeNode;
import vrml.external.field.*;
import vrml.external.exception.*;
import vrml.external.Node;
import vrml.external.Browser;
import vrml.field.*;
import vrml.node.*;
import com.ms.security.*;
import com.ms.security.permissions.*;

public class VRMLObject extends Object implements EventOutObserver
{
    // Browser we're using
    Browser browser;

    // Root of the scene graph (to which we add our nodes)
    Node Scene;
    Node [] shape;
    Node [] appearance;
    Node [] material;
    Node [] geometry;
    Node [] touch;
    Node [] tmr;
    Node [] Script;
    Node [] scaInt;
    Node [] posInt;
    Node [] oriInt;
    Node [] colInt;

    // EventIns of the root node
    public EventInMFNode addChildren;
    public EventInMFNode removeChildren;
    public EventInSFNode nodeIn;
    public EventInSFTime startTime;
    public EventInSFRotation Rotation;
    public EventInSFVec3f Translation;
    public EventInSFVec3f Scale;
    public EventInSFColor Color;
    public EventOutSFVec3f Scale_changed;
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
public EventOutSFRotation Rotation_changed;
public EventOutSFVec3f Translation_changed;
public EventOutSFColor Color_changed;
public EventOutSFTime TouchTime_changed;
public EventOutSFNode Node_name;
public EventOutSFBool isActive;

//Variables declaration
static int Object_ID;
int Object_No;
String ObjectName;
String TS;
String StringTrans;
String StringRot;
String StringScale;
String File;
public Node[] Object_array;
public Node[] Texture_array;

//--use for Animation--//
Node transform;
Node color;
Node LIGHT;
Node Scalar_Interpolator;
Node Position_Interpolator;
Node Orientation_Interpolator;
Node Color_Interpolator;
//------------------------//
Node node;
String ValueTrans;
String ValueRot;
String ValueColor;
String ValueScale;
float[] scale_after = new float[3];
float[] trans_after = new float[3];
float [] color_after = new float[3];
float[] rot_after = new float[4];
String TransRotScale;
static String name;
int xang;
int yang;
int zang;
Hashtable hashVRML = new Hashtable();
//Hashtable shopHT = new Hashtable();
boolean testrotflag=false;

//String definition for creating VRML object
final String Header =  "DEF ";
final String Header1 = " Transform { \n";
final String Translate = "  translation ";
final String Rotate = "\n"+ "rotation ";
final String Scaling = "\n"+ "scale ";
final String Children=  "\n"+
                        "children [ \n";
final String Shape0 = "Shape { \n";
final String Shape1 = "geometry ";
String Shape1a = "";
final String Shape2 = " { \n ";
String Shape3 = "";
final String Shape4 = " }\n";
final String Shape5 = "appearance Appearance { \n";
final String Shape6 = "DEF MAT";
final String Shape7 = " Material { \n";
final String Shape8 = "diffuseColor ";
final String Shape9 = "\n } \n";
final String Shape10 = "}\n}\n";

boolean activeSApply = false;
boolean activePApply = false;
boolean activeOApply = false;
ObjInfo curObject = null;

//String representation of boolean expressions
String strD_top = "";
String strD_side = "";
String strD_bottom = "";

//object properties
int pos; //object ID
String objName = ""; //name
int objType = 0; //type: cylinder, cone, sphere, box
String script = ""; //script

//Dimension
double D_radius = 1;
double D_height = 1;
double D_x = 1;
double D_y = 1;
double D_z = 1;
boolean D_top = true;
boolean D_side = true;
boolean D_bottom = true;

//position
double pos_x = 0;
double pos_y = 0;
double pos_z = 0;

//scale
double scale_x = 1;
double scale_y = 1;
double scale_z = 1;
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
//colour
double color_R = 0.2;
double color_G = 0.3;
double color_B = 0.1;

//rotation
double rot_x = 0;
double rot_y = 0;
double rot_z = 0;
double rot_angle = 0;

//text
String text = "";

//--------------------//
//*** Constructor ***
//--------------------//
VRMLObject(Browser browser, ObjInfo currentObj)
{
    this.browser = browser;
    Object_No = currentObj.pos;//Object #1: Ball
    ObjectName = currentObj.objName;

    curObject = currentObj;

    objName = curObject.objName;
    objType = curObject.objType;
    D_radius = curObject.D_radius;
    D_height = curObject.D_height;
    D_x = curObject.D_x;
    D_y = curObject.D_y;
    D_z = curObject.D_z;
    D_top = curObject.D_top;
    D_side = curObject.D_side;
    D_bottom = curObject.D_bottom;
    pos_x = curObject.pos_x;
    pos_y = curObject.pos_y;
    pos_z = curObject.pos_z;
    rot_x = curObject.rot_x;
    rot_y = curObject.rot_y;
    rot_z = curObject.rot_z;
    rot_angle = curObject.rot_angle;
    scale_x = curObject.scale_x;
    scale_y = curObject.scale_y;
    scale_z = curObject.scale_z;
    color_R = curObject.color_R;
    color_G = curObject.color_G;
    color_B = curObject.color_B;
    text = curObject.text;

    //change the boolean values to string
    strD_top = String.valueOf(D_top).toUpperCase();
    strD_side = String.valueOf(D_side).toUpperCase();
    strD_bottom = String.valueOf(D_bottom).toUpperCase();

    //check the RGB values
    if (color_R >= 1) {
        color_R = 1;
    }
    if (color_G >= 1) {
        color_G = 1;
    }
    if (color_B >= 1) {
        color_B = 1;
    }

    switch (objType) {
    case 0: Shape1a = "DEF " + objName + "Shape Cylinder";
            Shape3 = "radius " + D_radius + "\n" + "height " + D_height +
                        "\n" + "side " + strD_side + "\n" + "bottom " +
                        strD_bottom + "\n" + "top " + strD_top + "\n";
            break;
    case 1: Shape1a = "DEF " + objName + "Shape Cone";
            Shape3 = "bottomRadius " + D_radius + "\n" + "height " + D_height +
                        "\n" + "side " + strD_side + "\n" + "bottom " +
                        strD_bottom + "\n";
            break;
    case 2: Shape1a = "DEF " + objName + "Shape Sphere";
            Shape3 = "radius " + D_radius + "\n" ;
            break;
    case 3: Shape1a = "DEF " + objName + "Shape Box";
            Shape3 = "size " + D_x + " " + D_y + " " + D_z + "\n";
            break;
    case 4: Shape1a = "DEF " + objName + "Shape Text";
            Shape3 = "string [\"" + text + "\"]\n";
            break;
    }

    TS = "TS" + objName;

    String ObjectVRML = Header + objName + Header1 + Translate +
                        pos_x + " " + pos_y + " " + pos_z + Rotate +
                        rot_x + " " + rot_y + " " + rot_z + " " + rot_angle +
                        Scaling + scale_x + " " + scale_y + " " + scale_z +
                        Children + /**Body0 + PS + Body1 + Body3 +
                        Body0 + SS + Body2 + Body3 + **/ End ;

    String geomString = Shape1a + Shape2 + Shape3 + Shape4;

    String matString =  Shape6 + objName + Shape7 + Shape8 + color_R +
                        " " + color_G + " " + color_B + Shape9;
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
        String tmrString = "DEF tmr" + objName + " TimeSensor {cycleInterval 1 "
                           + "loop FALSE}";

        String scalarInt = "DEF SI" + objName + " ScalarInterpolator { " +
                           "key [0, 0.2, 0.4, 0.6, 0.8, 1 ] " +
                           "keyValue [0, 0.5, 0, 1, 0.5, 0  ] } "  ;

        String positionInt = "DEF PI" + objName + " PositionInterpolator { " +
                           "key [0, 0.2, 0.4, 0.6, 0.8, 1 ] " +
                           "keyValue [   1 4 0, 1 2 0, -1 2 0, -1 4 0, 1 2 0, 1 4 0] } "  ;

        String orientationInt = "DEF OI" + objName + " OrientationInterpolator { " +
                           "key [0, 0.2, 0.4, 0.6, 0.8, 1 ] " +
                           "keyValue [0 1 0 0,  0 1 0 1.57,  0 1 0 3.14, 0 1 0 4.71, 0 1 0, 5, 0 1 0 6.28
] } "  ;

        String colorInt = "DEF CI" + objName + " ColorInterpolator { " +
                           "key [0, 0.2, 0.4, 0.6, 0.8, 1 ] " +
                           "keyValue [ 1 0 0, 0 1 0, 0 0 1, 0 1 0, 1 0 0, 1 1 0] } "  ;

        //Displaying the product into Scene
        Object_array = browser.createVrmlFromString(ObjectVRML);
        transform = Object_array[0];

        // Make the shape node and its children
        shape = browser.createVrmlFromString("Shape {}");
        appearance = browser.createVrmlFromString("Appearance {}");

        material = browser.createVrmlFromString(matString);
        geometry = browser.createVrmlFromString(geomString);
        touch = browser.createVrmlFromString("DEF " + TS + " TouchSensor { } ");
        addChildren = (EventInMFNode) transform.getEventIn("addChildren");
        removeChildren = (EventInMFNode) transform.getEventIn("removeChildren");

        //make timer node
        tmr = browser.createVrmlFromString(tmrString);
        addChildren.setValue(tmr);

        //make interpolator nodes
        scaInt = browser.createVrmlFromString(scalarInt);
        addChildren.setValue(scaInt);

        posInt = browser.createVrmlFromString(positionInt);
        addChildren.setValue(posInt);

        oriInt = browser.createVrmlFromString(orientationInt);
        addChildren.setValue(oriInt);

        colInt = browser.createVrmlFromString(colorInt);
        addChildren.setValue(colInt);

        // Add the material to the appearance
        nodeIn = (EventInSFNode) appearance[0].getEventIn("set_material");
        nodeIn.setValue(material[0]);

        // Add the appearance to the shape
        nodeIn = (EventInSFNode) shape[0].getEventIn("set_appearance");
        nodeIn.setValue(appearance[0]);

        // Add the geometry to the shape
        nodeIn = (EventInSFNode) shape[0].getEventIn("set_geometry");
        nodeIn.setValue(geometry[0]);

        // Add the shape to the transform
        addChildren.setValue(shape);

        // Add touchsensor to transform
        addChildren.setValue(touch);


        //sending the EventOut and EventIn
        Translation = (EventInSFVec3f) transform.getEventIn("translation");
        Rotation = (EventInSFRotation) transform.getEventIn("rotation");
        Scale = (EventInSFVec3f) transform.getEventIn("scale");
        Color = (EventInSFColor) material[0].getEventIn("diffuseColor");
        startTime = (EventInSFTime) tmr[0].getEventIn("startTime");
        Scale_changed = (EventOutSFVec3f) transform.getEventOut("scale");
        Translation_changed = (EventOutSFVec3f) transform.getEventOut("translation");
        Rotation_changed = (EventOutSFRotation) transform.getEventOut("rotation");
        Color_changed = (EventOutSFColor) material[0].getEventOut("diffuseColor");
        TouchTime_changed = (EventOutSFTime) touch[0].getEventOut("touchTime");

        // for sending the "TOUCH SENSOR" event to observer
        TouchTime_changed.advise(this,null);
        node = browser.getNode("ROOT");

        Scalar_Interpolator = scaInt[0];
        Position_Interpolator = posInt[0];
        Orientation_Interpolator = oriInt[0];
        Color_Interpolator = colInt[0];

        Node[] trans_nodes = new Node[4];
        trans_nodes[0] = Scalar_Interpolator;
        trans_nodes[1] = Position_Interpolator;
        trans_nodes[2] = Orientation_Interpolator;
        trans_nodes[3] = Color_Interpolator;
        addChildren.setValue(trans_nodes);
        //-------------------------//

        //ROUTE statements
        browser.addRoute(tmr[0], "fraction_changed", Position_Interpolator, "set_fraction");
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
    browser.addRoute(tmr[0], "fraction_changed", Orientation_Interpolator, "set_fraction");
    browser.addRoute(tmr[0], "fraction_changed", Color_Interpolator, "set_fraction");
    browser.addRoute(Position_Interpolator, "value_changed", transform, "set_translation");
    browser.addRoute(Orientation_Interpolator, "value_changed", transform, "set_rotation");
    browser.addRoute(Color_Interpolator, "value_changed", material[0], "set_diffuseColor");

    trans_after = Translation_changed.getValue();
    rot_after = Rotation_changed.getValue();
    scale_after = Scale_changed.getValue();
    color_after = Color_changed.getValue();
    ValueTrans = "";
    ValueRot = "";
    ValueScale="";
    ValueColor = "";
    for ( int i = 0; i < 3; i++ )
    {
        ValueTrans = ValueTrans + " ";
        ValueTrans = ValueTrans + Float.toString(trans_after[i]);
    }
    for ( int i = 0; i < 4; i++ )
    {
        ValueRot = ValueRot + " ";
        ValueRot = ValueRot + Float.toString(rot_after[i]);
    }
    for ( int i = 0; i < 3; i++ )
    {
        ValueScale = ValueScale + " ";
        ValueScale = ValueScale + Float.toString(scale_after[i]);
    }
    for ( int i = 0; i < 3; i++ )
    {
        ValueColor = ValueColor + " ";
        ValueColor = ValueColor + Float.toString(color_after[i]);
    }
    //      TransRotScale = ValueTrans + ValueRot+ ValueScale;
    //      curObject.ValueTrans = ValueTrans;
    //      curObject.ValueRot = ValueRot;
    //      curObject.ValueScale = ValueScale;
}
// callback method is illustrating sending/
// receiving of events and registering
// of callbacks on EventOuts.***
public void callback(EventOut event, double time, Object userData)
{
    curObject.isClicked();
}
//  Method 2: get ID/ObjectName info***
public static String getCallBackName()
{
    return name;
}

public String getName()
{
    return ObjectName;
}

public static int getCallBackID()
{
    return Object_ID;
}
public int getObjectID()
{
    return Object_No;
}
// start Animation by addRoute into browser***
public void startAnimation(VRMLBuilder vrmlBuild)
{
    int field_type;
    boolean result;
    String Field;
    String now;

    //start animation timer
    Field = "startTime";
    field_type = vrmlBuild.getFieldType(tmr[0],Field);
    now = String.valueOf(System.currentTimeMillis()/1000 );
    result = vrmlBuild.changeEAI(tmr[0], Field, new MFValue(field_type, now));

}

// Method 4: stop Animation by addRoute into browser***
public void stopAnimation(String interpolatorType)
{
    if (interpolatorType.equals("Position"))
    {
        browser.deleteRoute(Position_Interpolator, "value_changed", transform, "set_translation");

    }else
    if (interpolatorType.equals("Orientation"))
    {
        browser.deleteRoute(Orientation_Interpolator, "value_changed", transform, "set_rotation");
    }
    if (interpolatorType.equals("Color"))
    {
        browser.deleteRoute(Color_Interpolator, "value_changed", material[0], "set_diffuseColor");
    }
}

public void valuesChanged()
{
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
        D_radius = curObject.D_radius;
        D_height = curObject.D_height;
        D_x = curObject.D_x;
        D_y = curObject.D_y;
        D_z = curObject.D_z;
        D_top = curObject.D_top;
        D_side = curObject.D_side;
        D_bottom = curObject.D_bottom;
        pos_x = curObject.pos_x;
        pos_y = curObject.pos_y;
        pos_z = curObject.pos_z;
        rot_x = curObject.rot_x;
        rot_y = curObject.rot_y;
        rot_z = curObject.rot_z;
        rot_angle = curObject.rot_angle;
        scale_x = curObject.scale_x;
        scale_y = curObject.scale_y;
        scale_z = curObject.scale_z;
        color_R = curObject.color_R;
        color_G = curObject.color_G;
        color_B = curObject.color_B;
        text = curObject.text;

        script = curObject.script;

        objType = curObject.objType;

        switch (objType) {
            case 0: Shape1a = "DEF " + objName + "Shape Cylinder";
                    Shape3 = "radius " + D_radius + "\n" + "height " + D_height +
                                "\n" + "side " + strD_side + "\n" + "bottom " +
                                strD_bottom + "\n" + "top " + strD_top + "\n";
                    break;
            case 1: Shape1a = "DEF " + objName + "Shape Cone";
                    Shape3 = "bottomRadius " + D_radius + "\n" + "height " + D_height +
                                "\n" + "side " + strD_side + "\n" + "bottom " +
                                strD_bottom + "\n";
                    break;
            case 2: Shape1a = "DEF " + objName + "Shape Sphere";
                    Shape3 = "radius " + D_radius + "\n" ;
                    break;
            case 3: Shape1a = "DEF " + objName + "Shape Box";
                    Shape3 = "size " + D_x + " " + D_y + " " + D_z + "\n";
                    break;
            case 4: Shape1a = "DEF " + objName + "Shape Text";
                    Shape3 = "string [\"" + text + "\"]\n";
                    break;
        }

        String geomString = Shape1a + Shape2 + Shape3 + Shape4;
        geometry = browser.createVrmlFromString(geomString);

        nodeIn = (EventInSFNode) shape[0].getEventIn("set_geometry");
        nodeIn.setValue(geometry[0]);

    }

    public void getState()
    {
        //get the 10 values of the object
        trans_after = Translation_changed.getValue();
        rot_after = Rotation_changed.getValue();
        scale_after = Scale_changed.getValue();
        color_after = Color_changed.getValue();
        ValueTrans = "";
        ValueRot = "";
        ValueScale="";
        ValueColor="";
        for ( int i = 0; i < 3; i++ )
        {
            ValueTrans = ValueTrans + " ";
            ValueTrans = ValueTrans + Float.toString(trans_after[i]);
        }
        for ( int i = 0; i < 4; i++ )
        {
            ValueRot = ValueRot + " ";
            ValueRot = ValueRot + Float.toString(rot_after[i]);
        }
        for ( int i = 0; i < 3; i++ )
        {
            ValueScale = ValueScale + " ";
            ValueScale = ValueScale + Float.toString(scale_after[i]);
        }
        for ( int i = 0; i < 3; i++ )
        {
            ValueColor = ValueColor + " ";
            ValueColor = ValueColor + Float.toString(color_after[i]);
        }
        TransRotScale = ValueTrans + ValueRot+ ValueScale;
    }

}
```

//--------------clientConnect.java--------------//

```
import java.io.*;
import java.net.*;

//permissions for file io
import java.security.*;
import com.ms.security.*;
import com.ms.security.permissions.*;
```

ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

```
public class ClientConnect
{
    Socket socket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader fromServer = null;

    //constructor
    public ClientConnect()
    {

    }

    public boolean uploadFile(File f, String ipAddress, String port)
    {
        try
        {
            Integer intPort = Integer.valueOf(port);
            PolicyEngine.assertPermission(PermissionID.FILEIO);
            socket = new Socket(ipAddress, intPort.intValue());
            out = new PrintWriter(socket.getOutputStream(), true);
            in = new BufferedReader(new FileReader(f));
            fromServer = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String line = "";
            String serverMsg ="";

            while ((serverMsg = fromServer.readLine()) != null)
            {
                System.out.println("Server: " + serverMsg);
                if (serverMsg.trim().equals("SUCCESS"))
                {
                    return true;
                } else if (serverMsg.trim().equals("LALALA"))
                {
                    return false;
                }

                //send filename first
                //**NOTE: need to generate new filename at server side later!
                out.println("FILENAME " + f.getName());
                out.flush();

                //send file contents
                while ((line = in.readLine()) != null)
                {
                    out.println(line.trim());
                    out.flush();
                }

                //send message indicating end of file
                out.println("END");
                out.flush();

            }

            out.close();
            in.close();
            socket.close();
        } catch (UnknownHostException e)
        {
            System.out.println("Don't know about host: " + ipAddress);
            return false;
        } catch  (IOException e)
        {
            System.out.println("Couldn't get I/O for the connection to: " + ipAddress);
            return false;
        }
        return false;
    }
}
```